Theoretical Computer Science 126 (1994) 113–142
Elsevier

Finite tree automata with cost functions

Helmut Seidl*

Fachbereich Informatik, Universität des Saarlandes, Postfach 1150, D-66041 Saarbrücken, Germany

Abstract

Seidl, H., Finite tree automata with cost functions, Theoretical Computer Science 126 (1994) 113–142.

Cost functions for tree automata are mappings from transitions to (tuples of) polynomials over some semiring. We consider four semirings, namely, N the semiring of nonnegative integers, A the "arctical semiring", T the tropical semiring and F the semiring of finite subsets of nonnegative integers. We show: for semirings N and A it is decidable in polynomial time whether or not the costs of accepting computations is bounded; for F it is decidable in polynomial time whether or not the *cardinalities* of occurring cost sets are bounded. In all three cases, we derive explicit upper bounds. For semiring T, we prove decidability of boundedness as well, but obtain a polynomial-time algorithm only in case that the degrees of occurring polynomials are at most 1.

For N and A, we extend our results to multidimensional cost functions.

0. Introduction

Finite tree automata are finite-state devices which operate on labeled ordered trees. *Cost functions* c for tree automata map transitions to (tuples of) polynomials over some semiring R such that every computation obtains a *cost* $c(\phi)$ in R (resp. R^d in the multidimensional case). A pair of a finite tree automaton and a cost function is called a *cost automaton*. There are two reasons why we are interested in finite tree automata with cost functions.

First, finite tree automata are an important tool of compiler generating systems like OPTRAN where they are applied for generating code selectors from descriptions of target machine assembly languages [2, 4, 12]. A generated tree automaton A is meant to traverse the abstract syntax tree of an input program. The different accepting

Correspondence to: H. Seidl, Fachbereich Informatik, Universität des Saarlandes, Postfach 1150, D-66041 Saarbrücken, Germany.

* Partially supported by ASMICS and Deutsche Forschungsgemeinschaft, SFB 124 TB-C1.

0304-3975/94/$07.00 © 1994—Elsevier Science B.V. All rights reserved
SSDI 0304-3975(93)E0126-O

computations of A correspond to different possibilities of target machine code generation. From these, the "cheapest" one is selected according to some suitable cost measure. In fact, these measures are of a type similar to the cost functions we consider here.

The second reason is of a more theoretical nature. In [1] Courcelle and Mosbah investigated MS (i.e. monadic second-order) evaluations on graphs and came up with a general method to translate these to evaluations of graph expressions over suitable semirings. Their method allows the derivation of polynomial algorithms for a huge class of problems on families of graphs like series parallel graphs, graphs definable by context-free hyperedge replacement grammars, etc. To be more specific, consider e.g. the family $\mathrm{SP}(\Sigma)$ of *series parallel graphs* with edge labels from Σ. It consists of acyclic digraphs with one source and one sink whose edges are labeled by elements from Σ. $\mathrm{SP}(\Sigma)$ can be defined inductively as follows:

(1) A single directed edge e labeled by some $a\in\Sigma$ is in $\mathrm{SP}(\Sigma)$;

(2) If $G_1, G_2\in\mathrm{SP}(\Sigma)$ then both the series and the parallel composition of G_1 and G_2 are elements from $\mathrm{SP}(\Sigma)$. The series composition is obtained from G_1 and G_2 by pasting the sink of G_1 with the source of G_2 whereas the parallel composition is obtained by pasting the two sources as well as the two sinks.

Every series parallel graph can be represented by a graph expression, i.e. some tree over the ranked alphabet $\Sigma\cup\{\|,\cdot\}$ where symbols $a\in\Sigma$ represent edges labeled by a, $\cdot$ and $\|$ have rank 2 and denote series and parallel composition, respectively.

Let G denote the graph represented by the graph expression g (the binary operators written in infix notation):

$$g=a\|((a\cdot d)\|b)\cdot c).$$

If we are interested in the *number* of paths from the source to the sink of G, we use an evaluation of g over the semiring $\boldsymbol{N}$ of naturals. Namely, the leaves of the expression tree (i.e. the edges of G) are counted 1, whereas $\|$ and $\cdot$ are viewed as the polynomials x_1+x_2 and x_1x_2. Evaluating g, we obtain 3 as a result.

To compute the *length* of the shortest path from the source to the sink of G, we employ the *tropical* semiring $\boldsymbol{T}$ where addition is $\sqcap$ and multiplication is $+$. We attach cost one to the leaves; to $\|$ and $\cdot$ we attach the polynomials $x_1\sqcap x_2$ and x_1+x_2, respectively. The result is 1.

The maximal *breadth* of G is computed over the "arctical" semiring $\boldsymbol{A}$. Here, addition is $\sqcup$ instead of $\sqcap$ whereas multiplication is again $+$. As above, cost one is attached to the leaves of the expression, to $\|$ we now attach the polynomial x_1+x_2, and to $\cdot$ the polynomial $x_1\sqcup x_2$. Thus, evaluation of g yields 3.

Finally, to compute the *set of path lengths* of G, we need semiring $\boldsymbol{F}$ which consists of all finite subsets of $\mathbb{N}_0$ where addition is set union and multiplication is addition of numbers extended to sets. We attach cost $\{1\}$ to the leaves of the expression. $\|$ is interpreted as the polynomial $x_1\cup x_2$, and $\cdot$ as the polynomial x_1+x_2. The resulting set is $\{1,2,3\}$. We are interested in decision problems like the following:

- Is there a global bound to the breadth of graphs described by graph expressions from some tree language L?
- Is there a global bound to the length of the shortest path in graphs described by graph expressions from some tree language L?

Provided the tree language L is the language accepted by some finite tree automaton A, the evaluation functions in question turn out to be cost functions for A over a suitable semiring $\boldsymbol{R}$. Thus, decision problems of the above type correspond to boundedness problems for tree automata with cost functions over $\boldsymbol{R}$. In [5], decidability was proved for the case of cost functions over $\boldsymbol{N}$ or $\boldsymbol{A}$. We improve their results by giving *polynomial-time* algorithms for boundedness in these cases. Our algorithms are based on *syntactic* properties which are easy to test but which (at least for "parameter-reduced" automata) precisely characterize boundedness. Furthermore, we consider semirings $\boldsymbol{T}$ and $\boldsymbol{F}$. In order to derive upper bounds in case of semiring $\boldsymbol{T}$, we introduce a new technique to pump up different (even overlapping) subparts of a computation simultaneously. These upper bounds allow the construction of a polynomial-time algorithm (at least), provided the degrees of occurring polynomials are at most 1. In case of $\boldsymbol{F}$, we show that it can be decided in polynomial time whether or not the *cardinalities* of cost sets of accepting computations are bounded or not. This polynomial-time algorithm is again based on a structural characterization of boundedness. Additionally, it makes use of the corresponding decision procedure for semiring $\boldsymbol{A}$ and a subroutine which decides whether or not the cardinalities of occurring cost sets are at most 1.

The paper is organized as follows. In Section 1, we introduce our basic notation concerning trees and tree automata. In Section 2, we introduce semirings and notation concerning polynomials over semirings and cost functions. We give a general method to parameter-reduce $\boldsymbol{R}$-cost automata (Theorem 2.3). In Section 3, we consider the case of 1-dimensional cost functions for each of the semirings $\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{T}$ and $\boldsymbol{F}$ (Theorems 3.2, 3.4, 3.5 and 3.19). For semirings $\boldsymbol{N}, \boldsymbol{A}$ and $\boldsymbol{F}$, we derive structural characterizations of boundedness which are decidable in polynomial time. For all four semirings, we provide explicit upper bounds depending only on structural parameters of the automata and cost functions in question. Section 4 extends the results of Sections 2 and 3 to multidimensional cost functions for the semirings $\boldsymbol{N}$ and $\boldsymbol{A}$ (Theorems 4.1, 4.4 and 4.6). Section 5 concludes with open problems.

The present paper is an extended version of [11]. It adds details about proofs and extends the boundedness result for $\boldsymbol{T}$ to the case of polynomials of arbitrary degree.

1. Basics

A *ranked alphabet* or *signature* is a pair (Σ, ρ) where Σ is a finite alphabet and $\rho \cdot \Sigma \to \mathbb{N}_0$ is a function mapping symbols to their ranks. Usually, if ρ is understood, we

write Σ for short and define $\Sigma_j = \rho^{-1}(j)$. The maximal j such that $\Sigma_j \neq \emptyset$ is called the *rank* of Σ.

T_Σ denotes the free Σ-algebra of (finite ordered Σ-labeled) *trees*, i.e. T_Σ is the smallest set T satisfying (i) $\Sigma_0 \subseteq T$, and (ii) if $a \in \Sigma_m$ and $t_1, \ldots, t_m \in T$ then $a(t_1, \ldots, t_m) \in T$.

Note: (i) can be viewed as the subcase of (ii) where $m = 0$.

The set of *nodes* of t, $O(t)$ is the subset of $\mathbb{N}^*$ defined by $O(t) = \{\varepsilon\} \cup \bigcup_{j=1}^{m} j \cdot O(t_j)$, where $t = a(t_1, \ldots, t_m)$ for some $a \in \Sigma_m$, $m \geq 0$. The *size* $|t|$ of t is defined as the cardinality of $O(t)$. For nodes r, r' we write $r \leq r'$ ($r < r'$) to denote that r is a (proper) prefix of r'. If neither $r \leq r'$ nor $r' < r$ then we call r and r' *incomparable* and write $r \asymp r'$. t defines maps $t(\_): O(t) \to \Sigma$ and $t/\_: O(t) \to T_\Sigma$ mapping the nodes o of t to their labels or the subtree of t with root o, respectively. We have

$$t(o) = \begin{cases} a & \text{if } o = \varepsilon \\ t_j(o') & \text{if } o = j \cdot o' \end{cases} \quad \text{and} \quad t/o = \begin{cases} t & \text{if } o = \varepsilon \\ t_j/o' & \text{if } o = j \cdot o' . \end{cases}$$

Let X denote a set of variables of rank 0. Define $T_\Sigma(X) = T_{\Sigma \cup X}$. We use this different notation in order to indicate which variables are to be substituted. (Clearly, $T_\Sigma \subseteq T_\Sigma(X)$.) Assume $t \in T_\Sigma(X)$. t is called *X-proper* iff every $x \in X$ occurs in t exactly once. If $X = \{x\}$ we write *x-proper* instead of *$\{x\}$-proper*, and if X is understood we skip the prefix X.

Every map $\theta: X \to T_\Sigma(X)$ can be extended to a map $\theta: T_\Sigma(X) \to T_\Sigma(X)$ by $t\theta = a(t_1\theta, \ldots, t_m\theta)$ whenever $t = a(t_1, \ldots, t_m)$ with $a \in \Sigma$. θ is called *X-substitution* or simply substitution if X is understood. If $X = \{x_1, \ldots, x_m\}$ and $x_i\theta = t_i$, we denote $t\theta$ also by $t[t_1, \ldots, t_m]$. Of special importance is the case where the set X of variables which are to be substituted consists of just one element x. Assume $x\theta = t_2$ and $t_1 \in T_\Sigma(x) = T_\Sigma(\{x\})$. Then we write $t_1\theta = t_1 t_2$. The set $T_\Sigma(x)$ is a monoid w.r.t. x-substitution. (The neutral element is x.)

For $t \in T_\Sigma(X)$, we define the kth power t^k of t by $t^1 = t$ and for $k > 1$, $t^k = t\theta$ where $x\theta = t^{k-1}$ for every $x \in X$. Observe that even if t was proper, t^k does not need to be proper as well.

A *finite tree automaton* (FTA for short) is a 4-tuple $A = (Q, \Sigma, \delta, Q_F)$ where Q is a finite set of *states*, $Q_F \subseteq Q$ is the set of *final* states, Σ is the signature of *input trees*, and $\delta \subseteq \bigcup_{m \geq 0} Q \times \Sigma_m \times Q^m$ is the set of *transitions* of A; the transitions in $\delta \cap \bigcup_{m \geq 0} \{q\} \times \Sigma_m \times Q^m$ are also called *q-transitions*.

For Sections 1–3, let X denote the set of variables $\{x_j \mid j \in \mathbb{N}\}$, and X_k the set of variables $\{x_j \mid j \in [1, k]\}$ for $k \in \mathbb{N}$. Let $t = a(t_1, \ldots, t_m) \in T_\Sigma(X_k)$ and $q, q_1, \ldots, q_k \in Q$. A *$(q, q_1 \ldots q_k)$-computation* ϕ of A for t starts at variables x_j in states q_j and consists of $(p_j, q_1 \ldots q_k)$-computations of A for the subtrees t_j, $j = 1, \ldots, m$, together with a transition $(q, a, p_1 \ldots p_m) \in \delta$ for the root. We write the state at the root to the left of the states at the variable leaves. This convention is chosen in accordance with our prefix notation of trees and the left-to-right order of substitutions. Formally, we represent ϕ as a tree over signature δ and set of variables X_k as follows. If $t = x_j$ and $q = q_j$ then $\phi = x_j$. If $t = a(t_1, \ldots, t_m)$ then $\phi = \tau(\phi_1, \ldots, \phi_m)$ where $\tau = (q, a, p_1 \ldots p_m) \in \delta$ for suitable states $p_1, \ldots, p_m \in Q$ and ϕ_j is a $(p_j, q_1 \ldots q_k)$-computation for t_j, $j = 1, \ldots, m$.

The set of all $(q, q_1 \dots q_k)$-computations is denoted by $\Phi^{(q, q_1 \dots q_k)}$. Assume $t \in T_\Sigma(X_k)$ and $t = t_0[t_1, \dots, t_k]$. Assume ϕ_0 is a $(q, p_1 \dots p_k)$-computation for t_0, and ϕ_i are $(p_i, q_1 \dots q_m)$-computations for t_i, $i = 1, \dots, k$. Then $\phi_0[\phi_1, \dots, \phi_k]$ is a $(q, q_1 \dots q_m)$-computation ϕ of A for t. Conversely, if t_0 contains exactly one occurrence of any x_j, $j = 1, \dots, k$ (i.e. is X_k-proper), then every $(q, q_1 \dots q_m)$-computation ϕ for t can be uniquely decomposed into a $(q, p_1 \dots p_k)$-computation ϕ_0 for t_0, and $(p_i, q_1 \dots q_m)$-computations ϕ_i for t_i (for suitable states p_i) such that $\phi = \phi_0[\phi_1, \dots, \phi_k]$. ϕ_i is called *subcomputation* of ϕ on t_i.

A (q, ε)-computation is also called q-computation. A q-computation is called *accepting* iff $q \in Q_F$. $L(A) = \{t \in T_\Sigma \mid$ there is an accepting computation of A for $t\}$ is the *language* accepted by A. The *size* of A, $|A|$, is defined by $|A| = \sum_{(q, a, q_1 \dots q_m) \in \delta} (m+2)$. For estimating the complexities of our algorithms, we always assume that the input signature Σ is *fixed*. Only the sets of states and transitions vary. Thus, especially, the rank of Σ is viewed as a *constant*.

The algorithms for tree automata we consider mainly run on a data structure called the *trace graph*. For FTA $A = (Q, \Sigma, \delta, Q_F)$ the *trace graph* of A is the edge-labeled digraph $G(A) = (V, E)$ where the set of vertices V equals Q, and the set of edges E consists of all triples $(q, \langle \tau, j \rangle, q')$ where $\tau = (q, a, q_1 \dots q_m)$ is a q-transition in δ with $q_j = q'$.

Call a state $q \in Q$ *useless* if there is no accepting computation in which there occurs a q-transition, and *useful* otherwise. An FTA A is called *reduced* iff A has no useless states. Useless states can be removed without changing the "behavior" of A. We have now the following proposition.

Proposition 1.1. *For every* FTA $A = (Q, \Sigma, \delta, Q_F)$ *there is an* FTA $A_r = (Q_r, \Sigma, \delta_r, Q_{r,F})$ *with* $L(A) = L(A_r)$ *such that*

- $Q_r \subseteq Q$, $\delta_r \subseteq \delta$ *and* $Q_{r,F} \subseteq Q_F$;
- A_r *is reduced.*

A_r *can be constructed from* A *in polynomial time.*

2. Cost automata

In this section, we consider polynomials over semirings and introduce cost automata. A (commutative) *semiring* $\boldsymbol{R}$ (with 0 and 1) is a 5-tuple $\boldsymbol{R} = (R, +, \cdot, 0, 1)$ where $(R, +, 0)$ and $(R, \cdot, 1)$ are commutative monoids with neutral elements 0 and 1, respectively, such that

$$0 \cdot a = 0 \text{ and } a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

for all $a, b, c \in R$. R is also called the *carrier* of $\boldsymbol{R}$. As usual, we also write $r \in \boldsymbol{R}$ iff r is an element of the carrier of $\boldsymbol{R}$. Especially, we consider the following four semirings.

(1) The *naturals* $\boldsymbol{N} = (\mathbb{N}_0, +, \cdot, 0, 1)$ with usual addition and multiplication;

(2) The *arctical semiring* $\boldsymbol{A}$ with carrier $\mathbb{N}_0 \cup \{-\infty\}$ where addition is $\sqcup$ and multiplication is $+$ on integers extended by $x + -\infty = -\infty + x = -\infty$;

(3) The *tropical semiring* $\boldsymbol{T}$ with carrier $\mathbb{N}_0 \cup \{\infty\}$ where addition is $\sqcap$ and multiplication is $+$ on integers extended by $x+\infty=\infty+x=\infty$;

(4) The semiring $\boldsymbol{F}$ whose carrier consists of all finite subsets of nonegative integers, where addition is set union, and multiplication is addition extended to sets i.e. $A+B=\{a+b \mid a\in A,\ b\in B\}$.

Observe that all four semirings have a natural (in case of $\boldsymbol{F}$ partial) ordering which, in case of $\boldsymbol{N}$, $\boldsymbol{T}$ and $\boldsymbol{A}$, is derived from the ordering $-\infty<0<1<2<\cdots<n<\cdots<\infty$. The ordering of $\boldsymbol{F}$ is given by set inclusion. For these orderings all finite least upper bounds exist. In $\boldsymbol{N}$, $\boldsymbol{A}$ and $\boldsymbol{T}$, we denote the least upper bound of an unbounded set of semiring elements by ∞. For $\boldsymbol{F}$, the least upper bound of a finite number of semiring elements is their union. A least upper bound of an infinite number of elements from $\boldsymbol{F}$ is defined as their union as well – although it may no longer be an element of $\boldsymbol{F}$.

The set of polynomials over $\boldsymbol{R}$ with variables from X is denoted by $\boldsymbol{R}[X]$. A *monomial* m is a polynomial of the form $h\cdot x_1^{\varepsilon_1}\cdot\ldots\cdot x_k^{\varepsilon_k}$. h is called the *coefficient*, and $\varepsilon_1+\cdots+\varepsilon_k$ the *degree* of m. The *size* of m, $|m|$ is defined as $|m|=1+\#\{j \mid \varepsilon_j\neq 0\}$. The latter definition refers to the intuition that both, every semiring element and every exponent can be stored in one storage cell of a random access machine (with uniform cost measure). For simplicity, we assume that such a machine can execute comparisons and semiring operations of two elements $r,r'\in\boldsymbol{R}$ in one step. Also, for every semiring homomorphism $H:\boldsymbol{R}\to\boldsymbol{R}'$ and $r\in\boldsymbol{R}$, $H(r)$ can be computed in one step. Thus, a polynomial algorithm (say, for a Turing machine) is only obtained from a polynomial algorithm for such a RAM if each involved comparison, semiring operation and application of homomorphism can be executed in polynomial time.

For our purposes, we assume that a polynomial $p\in\boldsymbol{R}[X]$ is always given as a (possibly empty) sum of monomials where no monomial has a coefficient 0 and for two monomials $m=h\cdot x_1^{\varepsilon_1}\cdot\ldots\cdot x_k^{\varepsilon_k}$ and $m'=h'\cdot x_1^{\varepsilon_1'}\cdot\ldots\cdot x_k^{\varepsilon_k'}$ of p, $\varepsilon_j\neq\varepsilon_j'$ for at least one j. Because of commutativity and associativity of "$+$", we consider two polynomials as equal whenever they contain the same set of monomials. Especially, $p=0$ iff the set of monomials of p is empty. Note that for semiring $\boldsymbol{R}=\boldsymbol{N}$, different polynomials also define different functions. This need not be the case for $\boldsymbol{R}\neq\boldsymbol{N}$.

The degree of polynomial p is the maximal degree of a monomial in p whereas we choose the *size* of p as the sum of the sizes of the monomials in p. The set of polynomials over $\boldsymbol{R}$ with variables from X of degree at most 1 is also denoted by $\boldsymbol{R}^{(1)}[X]$. Variable x_j *occurs* in p, or p *depends* on x_j iff p contains a monomial $h\cdot x_1^{\varepsilon_1}\cdot\ldots\cdot x_k^{\varepsilon_k}$ with $\varepsilon_j\neq 0$.

As for substitutions, a map $\theta:X\to\boldsymbol{R}[X]$ can be extended to a map $\boldsymbol{R}[X]\to\boldsymbol{R}[X]$ which is denoted by θ as well. For $f\in\boldsymbol{R}[X]$, $f\theta$ returns a polynomial representing the functional composition of the functions denoted by f and the $x_i\theta$. If $f\in\boldsymbol{R}$ then $f\theta=f$. If $f=x\in X$ then $f\theta=x\theta$. If $f=f_1*f_2$ for $*\in\{+,\cdot\}$ then $f\theta=(f_1\theta)*(f_2\theta)$. As for trees, we call θ an *X-substitution* or substitution, if X is understood. If $X=\{x_1,\ldots,x_m\}$ and $x_i\theta=f_i$, we denote $f\theta$ by $f[f_1,\ldots,f_m]$ and also write ff_1 for $f[f_1]$.

Fact 2.1. *Assume* $\boldsymbol{R}\in\{\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{F}\}$ *and* p *is a polynomial in* $\boldsymbol{R}[x]=\boldsymbol{R}[\{x\}]$. *Then the following holds*:

(1) *If* $b_1, b_2\in\boldsymbol{R}$ *and* $b_1\leqslant b_2$ *then* $p[b_1]\leqslant p[b_2]$.

(2) *Assume* x *occurs in* p *and* $b\in\boldsymbol{R}$.

- *If* $\boldsymbol{R}\subset\{\boldsymbol{N}, \boldsymbol{A}\}$, *then* $p[b]\geqslant b$.
- *If* $\boldsymbol{R}=\boldsymbol{F}$ *then* $\#p[b]\geqslant\#b$.

Assume $A=(Q, \Sigma, \delta, Q_F)$ is an FTA and $\boldsymbol{R}$ is a semiring. An *$\boldsymbol{R}$-cost function* for A is a mapping $\boldsymbol{c}:\delta\to\boldsymbol{R}[X]$ where $\boldsymbol{c}(\tau)\in\boldsymbol{R}[X_m]$ provided $\tau=(q, a, q_1\dots q_m)$. $\boldsymbol{c}(\_)$ is extended to computations ϕ in the natural way. If $\phi=x_j$ then $\boldsymbol{c}(\phi)=x_j$. If $\phi=\tau(\phi_1, \dots, \phi_m)$ for some $\tau\in\delta$ then $\boldsymbol{c}(\phi)=\boldsymbol{c}(\tau)[\boldsymbol{c}(\phi_1), \dots, \boldsymbol{c}(\phi_m)]$.

The following notion of a monomial expression of a computation is crucial when dealing with semirings $\boldsymbol{A}$ and $\boldsymbol{T}$. Let $M_{\boldsymbol{R}}[X]$ denote the set of monomials over $\boldsymbol{R}$ different from 0. A *monomial expression* (short form: expression) of a $(q, q_1\dots q_k)$-computation $\phi\neq x_j$ is a mapping $\sigma: O\to M_{\boldsymbol{R}}[X]$ with $O\subseteq O(\phi)$ such that:

(1) For $r\in O$, $\phi(r)=x_j$ implies $\sigma(r)=x_j$. Otherwise, $\sigma(r)$ is a monomial of $\phi(r)$.

(2) $\varepsilon\in O$. If $r\in O$ and $\sigma(r)=h\cdot\prod_{j=1}^{m} x_j^{\varepsilon_j}$ then $rj\in O$ iff $\varepsilon_j\neq 0$.

The set of all expressions of ϕ is denoted by $\mathrm{ME}(\phi)$. The domain of an expression σ is also referred to as $O(\sigma)$. The monomial $\sigma(r)$ is the monomial *chosen* by σ at r. For $r\in O(\sigma)$ define the cost $\boldsymbol{c}(r)$ inductively as follows: If $\phi(r)=x_j$ then $\boldsymbol{c}(r)=x_j$. Otherwise, if $\sigma(r)=h\cdot\prod_{j=1}^{m} x_j^{\varepsilon_j}$ then $\boldsymbol{c}(r)=h\cdot\prod_{\varepsilon_j\neq 0}\boldsymbol{c}(rj)^{\varepsilon_j}$.

Finally, let $\boldsymbol{c}(\sigma)=\boldsymbol{c}(\varepsilon)$. Our basic observation is that

$$\boldsymbol{c}(\phi)=\sum\{\boldsymbol{c}(\sigma)\mid\sigma\in\mathrm{ME}(\phi)\}.$$

Let $\sigma: O\to M_{\boldsymbol{R}}[X]$ be an expression of computation ϕ and $r\in O$. Then σ/r is the expression of ϕ/r with $O(\sigma/r)=\{r'\mid rr'\in O\}$ which is defined by $\sigma/r(r')=\sigma(rr')$. Accordingly, assume $\phi=\psi[\phi_1, \dots, \phi_k]$ for proper ψ such that the occurrences r_j of variables x_j are in O. Then σ can be decomposed into an expression σ_0 of ψ and expressions $\sigma_j=\sigma/r_j$ of ϕ_j. We write $\sigma=\sigma_0[\sigma_1, \dots, \sigma_k]$. Note that a similar decomposition may not be possible in case ψ is not proper since σ possibly contains different expressions for different occurrences of the same ϕ_j. In case not all r_j are in $O(\sigma)$, we also write $\sigma=\sigma_0[\sigma_1, \dots, \sigma_k]$ and define $\sigma_j=0$ whenever $r_j\notin O(\sigma)$.

An *$\boldsymbol{R}$-cost automaton* M is a pair $M=(A, \boldsymbol{c})$ where A is the FTA *underlying* M and $c:\delta\to\boldsymbol{R}[X]$ is an $\boldsymbol{R}$-cost function for A.

The *size* of M consists of the size of A together with the space to represent $\boldsymbol{c}$. Hence, we define $|M|=|A|+\sum_{\tau\in\delta}|\boldsymbol{c}(\tau)|$.

The set of costs of A w.r.t. $\boldsymbol{c}$, $\boldsymbol{c}(A)$ is defined by

$$\boldsymbol{c}(A)=\{\boldsymbol{c}(\phi)\mid\phi\text{ accepting computation of }A\}.$$

In case $\boldsymbol{R}\in\{\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{T}\}$, we are interested in whether the least upper bound of costs $\bigsqcup M=\bigsqcup\boldsymbol{c}(A)$ is finite. In case $\boldsymbol{R}=\boldsymbol{F}$, we are interested in whether the least upper bound of cost cardinalities $\#M=\bigsqcup\{\#B\mid B\in\boldsymbol{c}(A)\}$ is finite. If so, we say that the costs of M are *bounded* (in short, M is bounded).

We would like to eliminate costs of subcomputations which do not contribute to the final cost. Consider e.g. the semiring $\boldsymbol{N}$. A subcomputation may not contribute to the final cost if its cost is multiplied by 0. It turns out that we can decide "online" whether or not a given subcomputation has cost 0 (or, if we like, 1). In order to have a machinery general enough to cope with semirings $\boldsymbol{A}$, $\boldsymbol{T}$ and $\boldsymbol{F}$ as well, we introduce the notion of *faithful* subsets of semirings.

Assume $H: \boldsymbol{R} \to \boldsymbol{R}'$ is a surjective homomorphism of semirings where $\boldsymbol{R}'$ is finite. Subset $E \subseteq \boldsymbol{R}$ is called faithful (via H) iff $E \subseteq \{r \in \boldsymbol{R} \mid H^{-1}(H(r)) = \{r\}\}$.

Examples 2.2. For every $m \in \mathbb{N}$,

(1) $[0, m-1]$ is a faithful subset of $\boldsymbol{N}$,

(2) $\{-\infty\} \cup [0, m-1]$ is a faithful subset of $\boldsymbol{A}$,

(3) $[0, m-1] \cup \{\infty\}$ is a faithful subset of $\boldsymbol{T}$, and

(4) $2^{[0,m-1]}$ is a faithful subset of $\boldsymbol{F}$.

In all these four cases of faithful subsets $E \subseteq \boldsymbol{R}$, the subset E is faithful via some homomorphism $H_m: \boldsymbol{R} \to \boldsymbol{R}_m$ where $\boldsymbol{R}_m$ is a finite semiring obtained from $\boldsymbol{R}$ by an appropriate "truncation".

(1) $\boldsymbol{N}_m$ is the semiring with carrier $[0, m]$ where addition $+_m$ and multiplication $\cdot_m$ are defined by $x +_m y = m \sqcap (x + y)$ and $x \cdot_m y = m \sqcap (x \cdot y)$ such that $H_m(x) = x \sqcap m$.

(2) $\boldsymbol{A}_m$ is the semiring with carrier $\{-\infty\} \cup [0, m]$ ordered by $-\infty < 0 < 1 < \cdots < m$. As for $\boldsymbol{A}$, addition is $\sqcup$ and multiplication is $+_m$ defined by $x +_m y = m \sqcap (x + y)$ where "$+$" here is ordinary addition extended by $-\infty + x = x + -\infty = -\infty$. Again, $H_m(x) = x \sqcap m$.

(3) Analogous to $\boldsymbol{A}_m$, $\boldsymbol{T}_m$ is the semiring with carrier $[0, m] \cup \{\infty\}$ ordered by $0 < 1 < \cdots < m < \infty$. Addition is now $\sqcap$ and multiplication is $+_m$ defined by $x + \infty = \infty + x = \infty$, and $x +_m y = m \sqcap (x + y)$ whenever $x \neq \infty \neq y$. Now, $H_m(x) = \infty$ if $x = \infty$ and $H_m(x) = x \sqcap m$ otherwise.

(4) Finally, $\boldsymbol{F}_m$ is the semiring with carrier $2^{[0,m-1]} \cup \{u\}$ for some new symbol u. Addition in $\boldsymbol{F}_m$ is set union extended by $x \cup u = u \cup x = u$, whereas multiplication $+_m$ is defined by

$$A +_m B = \begin{cases} \{a + b \mid a \in A,\ b \in B\} & \text{if } a + b < m \text{ for all } a \in A,\ b \in B, \\ u & \text{otherwise.} \end{cases}$$

Now, $H_m(B) = B$ if $B \subseteq [0, m-1]$ and $H_m(B) = u$ otherwise.

Since all the semirings $\boldsymbol{R}_m$ are finite, we can incorporate the evaluation of cost functions $H_m\, \boldsymbol{c}$ into the computation of the tree automaton itself. Assume $M = (A, \boldsymbol{c})$ is an $\boldsymbol{R}$-cost automaton where A is reduced and $E \subseteq \boldsymbol{R}$ is faithful where $0 \in E$. M is called *E-reduced* iff there are sets $U_e(M)$, $e \in E$, such that a q-computation has cost e iff $q \in U_e(M)$.

We would like to use information about costs of subcomputations to "simplify" the polynomials involved. We call M *E-parameter-reduced* iff M is E-reduced and the following holds:

(1) For every $e \in E$ different from 0, $U_e(M) \subseteq Q_F$, and no state $q \in U_e(M)$ occurs as a successor state q_j in some transition $(q, a, q_1 \dots q_k) \in \delta$.

(2) If $\tau \in \delta$ is a q-transition with $q \in U_e(M)$ then $c(\tau) = e$.

(3) For every transition $\tau = (q, a, q_1 \dots q_k) \in \delta$, x_j does not occur in $c(\tau)$ iff $q_j \in U_0(M)$.

For our characterizations of bounded costs, we will refer only to E-parameter-reduced $\boldsymbol{R}$-cost automata where $E = \{0, 1\}$. For simplicity, we skip the prefix "E-" in this case.

Before we explain the corresponding results, we first convince ourselves that we may always assume (w.l.o.g.) that the cost automata in question are parameter-reduced. For this we prove Theorem 2.3.

Theorem 2.3. *Assume $\boldsymbol{R}$ is a semiring, and $E \subseteq \boldsymbol{R}$ is faithful via $H: \boldsymbol{R} \to \boldsymbol{R}'$ with $0 \in E$.*

(1) *For every $\boldsymbol{R}$-cost automaton $M = (A, c)$ there is an E-reduced $\boldsymbol{R}$-cost automaton $M_r = (A_r, c_r)$ such that*

- $L(A) = L(A_r)$ *and* $c(A) = c_r(A_r)$,
- *if n is the number of states of A and L the rank of the input signature then A_r has at most $(\# \boldsymbol{R}') \cdot n$ states and $|M_r| \leqslant |M| \cdot (\# \boldsymbol{R}')^L$,*
- M_r *can be constructed by a RAM in polynomial time.*

(2) *For every E-reduced $\boldsymbol{R}$-cost automaton $M = (A, c)$ there is an E-parameter-reduced $\boldsymbol{R}$-cost automaton $M_p = (A_p, c_p)$ such that*

- $L(A) = L(A_p)$ *and* $c(A) = c_p(A_p)$;
- *if A has n states then A_p has at most $2 \cdot n$ states and $|M_p| \leqslant 2 \cdot |M|$.*
- M_p *can be constructed by a RAM in polynomial time.*

Proof. The E-reduced cost automaton according to statement (1) is obtained from M simply by adding an extra component from $\boldsymbol{R}'$ to the states which records the image of the cost of a subcomputation under H. Formally, we construct a cost automaton $M_0 = (A_0, c_0)$ with $A_0 = (Q_0, \Sigma, \delta_0, Q_{0,F})$ where $Q_0 = Q \times \boldsymbol{R}'$ with $Q_{0,F} = Q_F \times \boldsymbol{R}'$ and δ_0 consists of all transitions $\langle \tau, \bar{r} \rangle = (\langle q, r \rangle, a, \langle q_1, r_1 \rangle \dots \langle q_k, r_k \rangle) \in \delta$ where $\tau = (q, a, q_1 \dots q_k) \in \delta$, $\bar{r} = (r_1, \dots, r_k) \in \boldsymbol{R}^k$ and $r = H' c(\tau)[r_1, \dots, r_k]$. For each such transition we put $c_0(\langle \tau, \bar{r} \rangle) = c(\tau)$.

Now, the E-reduced cost automaton M_r is obtained from M_0 by removing useless states and transitions and restricting the cost function c_0 to the set of remaining transitions.

Assume $M = (A, c)$ is an E-reduced cost automaton where $A = (Q, \Sigma, \delta, Q_F)$. The idea of the construction of an E-parameter-reduced cost automaton according to statement (2) is the following. We add a tag from $\{0, 1\}$ to the states of A, "0" means that the actual occurrence is situated in a subcomputation which finally is substituted into a variable *not* occurring in the polynomial in question. Opposed to that, costs generated at states marked "1" contribute to the final cost result. Formally, define a cost automaton $M_1 = (A_1, c_1)$, with $A_1 = (Q_1, \Sigma, \delta_1, Q_{1,F})$, by

$$Q_1 = Q \times \{0, 1\} \text{ with } Q_{1,F} = Q_F \times \{1\},$$

where δ_1 and c_1 are defined as follows.

Consider $\tau=(q, a, q_1 \dots q_k)\in\delta$. Then $\langle\tau,0\rangle=(\langle q,0\rangle, a, \langle q_1,0\rangle \cdots \langle q_k,0\rangle)\in\delta_1$ where $c_1(\langle\tau,0\rangle)=0$. If $q\in U_e(M)\cap Q_F$ for some $e\in E$ then also $\langle\tau,1\rangle=(\langle q,1\rangle, a, \langle q_1,0\rangle \cdots \langle q_k,0\rangle)\in\delta_1$ with $c(\langle\tau,1\rangle)=e$.

Furthermore, if $q\notin U_e(M)$ for any $e\in E$, define $p=c(\tau)[s_1,\dots,s_k]$ where $s_j=e_j$ if $q_j\in U_{e_j}(M)$ for $e_j\in E$ and $s_j=x_j$ otherwise. Then also $\langle\tau,1\rangle=(\langle q,1\rangle, a, \langle q_1,\beta_1\rangle \cdots \langle q_k,\beta_k\rangle)\in\delta_1$ where $\beta_j=1$ if x_j occurs in p and $\beta_j=0$ otherwise. We put $c_1(\langle\tau,1\rangle)=p$.

Finally, the E-parameter-reduced cost automaton according to statement (2) is obtained from M_1 by removing useless states and transitions and restricting c_1 to the set of remaining transitions. □

As a corollary we obtain the following.

Corollary 2.4. (1) *For $R\in\{N, T, A\}$, it can be decided for a given R-cost automaton $M=(A, c)$ and $m\in\mathbb{N}$ in time polynomial in $|M|$ and m, whether or not $\bigsqcup M<m$. For fixed $m\in\mathbb{N}$, it can be decided for a given F-cost automaton $M=(A, c)$ in polynomial time whether or not $c(A)\subseteq 2^{[0,m-1]}$.*

(2) *Let $R\in\{N, T, A, F\}$. For every R-cost automaton $M=(A, c)$, a parameter-reduced R-cost automaton $M_p=(A_p, c_p)$ can be constructed in polynomial time with $L(A)=L(A_p)$ and $c(A)=c_p(A_p)$.*

3. Upper bounds for bounded costs

In the next four subsections, we successively consider semirings N, A, T and F. In case of the semirings N, A and F, we present syntactical properties which characterize bounded costs of parameter-reduced cost automata and can be decided in polynomial time (Theorems 3.2, 3.4 and 3.19). For all four semirings N, A, T and F, we derive explicit upper bounds which for semiring T also proves decidability of boundedness (Theorem 3.5).

3.1. The semiring N

In this subsection, we consider costs in the semiring N.

Fact 3.1. *Assume $p\in N[x]$ and $b\notin\{0,1\}$. Then either $p[b]>b$ or $p\in N\cup\{x\}$.*

The N-cost automaton $M=(A, c)$ has property (N) iff

(N) For every edge $(q, \langle\tau, j\rangle, q')$ of a strong component of the trace graph $G(A)$, $c(\tau)=0$ or $c(\tau)=x_j$.

Obviously, it can be decided in polynomial time whether or not M has property (N). Now, we have the following theorem.

Theorem 3.2. *For a parameter-reduced* $\mathbf{N}$*-cost automaton* $M=(A, \boldsymbol{c})$ *the following three statements are equivalent:*

(1) M *is bounded,*

(2) M *has property* (N),

(3) $\bigsqcup M \leqslant \begin{cases} [(L+1)\cdot H]^n & \text{if } k=1, \\ [(L+1)^k \cdot H]^{k^n} & \text{if } k>1, \end{cases}$

where n *is the number of states of* A, L *is the rank of the input signature,* H *and* k *are, respectively, upper bounds for the coefficients and degrees of polynomials occurring in* $\boldsymbol{c}$. *It can be decided in polynomial time whether or not a given* $\mathbf{N}$*-cost automaton* M *is bounded.*

Proof. Assume M does not have property (N). Then there is an accepting computation $\phi=\psi_1\psi_2\psi_3$ for proper (f, q)-computation ψ_1, proper (q, q)-computation ψ_2 and q-computation ψ_3 such that $\boldsymbol{c}(\psi_2)\notin \boldsymbol{N}$ and $\boldsymbol{c}(\psi_2)\neq x_1$. Especially, it contains the variable x_1. It follows that $\boldsymbol{c}(\psi_3)\notin\{0, 1\}$ since M is parameter-reduced. Hence, by Fact 3.1 for every $k>1$, $\boldsymbol{c}(\psi_2^k\psi_3)=\boldsymbol{c}(\psi_2)^k\boldsymbol{c}(\psi_3)>k-1+\boldsymbol{c}(\psi_3)>k$. Since M is parameter-reduced, $\boldsymbol{c}(\psi_1)\notin \boldsymbol{N}$. Hence, by Fact 2.1(2), $\boldsymbol{c}(\psi_1\psi_2^k\psi_3)=\boldsymbol{c}(\psi_1)\boldsymbol{c}(\psi_2)^k\boldsymbol{c}(\psi_3)>k$, and the costs of M cannot be bounded. Therefore, (1) implies (2).

Since statement (3) trivially implies (1), it remains to show that (2) implies the upper bounds given in (3). So, assume M has property (N). We claim that for every accepting computation ϕ of M, some accepting computation ϕ' exists with $\boldsymbol{c}(\phi)=\boldsymbol{c}(\phi')$ such that every node $r\in O(\phi')$ has length $|r|<n$. Clearly, this claim implies the upper bounds given under (3).

To prove our claim, consider an accepting computation ϕ' with $\boldsymbol{c}(\phi)=\boldsymbol{c}(\phi')$ such that $\#O(\phi')$ is minimal under all such computations. For a contradiction, assume $r\geqslant n$ for some node $r\in O(\phi')$. Then $r=r_1r_2r_3$ for some $r_2\neq\varepsilon$ such that both $\phi'(r_1)$ and $\phi'(r_1r_2)$ are q-transitions for some state q. We can decompose ϕ' into $\phi'=\phi_1\phi_2\phi_3$ where $\phi_1=\phi'[x_1/r_1]$, $\phi_2=\phi'/r_1[x_1/r_2]$, and $\phi_3=\phi'/r_1r_2$. Since M has property (N), $\boldsymbol{c}(\phi_2)\in\{0, x_1\}$. Therefore, $\boldsymbol{c}(\phi')=\boldsymbol{c}(\phi_1\phi_3)$. Since $\phi_1\phi_3$ is an accepting computation and $\#O(\phi_1\phi_3)<\#O(\phi')$ this gives a contradiction. □

3.2. The semiring $\boldsymbol{A}$

Next, we consider costs in the semiring $\boldsymbol{A}$.

Fact 3.3. *Assume* $p\in\boldsymbol{A}[x]$ *and* $b\notin\{-\infty, 0\}$. *Then either* $p^k[b]>k$ *for all* $k>0$ *or* p *has one of the forms* $p=a$ *or* $p=a\sqcup x$ *for some* $a\in\boldsymbol{A}$.

The $\boldsymbol{A}$-cost automaton $M=(A, \boldsymbol{c})$ has property (A) iff

(A) for every edge $(q, \langle\tau, j\rangle, q')$ of a strong component of the trace graph $G(A)$, $\boldsymbol{c}(\tau)=-\infty$ or $\boldsymbol{c}(\tau)=p\sqcup x_j$ where $p\in\boldsymbol{A}[X\setminus\{x_j\}]$.

Obviously, it can be decided in polynomial time whether or not M has property (A). Now, we have the following theorem.

Theorem 3.4. *Assume $M=(A, \boldsymbol{c})$ is a parameter-reduced $\boldsymbol{A}$-cost automaton. The following three statements are equivalent:*

(1) *M is bounded,*

(2) *M has property* (A),

(3) $\bigsqcup M \leqslant \begin{cases} n \cdot H & \text{if } k=1, \\ 2Hk^n & \text{if } k>1, \end{cases}$

where n is the number of states of A, L is the rank of the input signature, H and k are, respectively, upper bounds for the coefficients and degrees of polynomials occurring in $\boldsymbol{c}$. It can be decided in polynomial time whether or not a given $\boldsymbol{A}$-cost automaton M is bounded.

Proof. The proof of implication (1)⇒(2) is analogous to the proof of the corresponding implication of the last theorem. Assume M does not have property (A). Then there is an accepting computation $\phi=\psi_1\psi_2\psi_3$ for proper (f, q)-computation ψ_1, proper (q, q)-computation ψ_2 and q-computation ψ_3 such that $\boldsymbol{c}(\psi_2)$ neither is in $\boldsymbol{A}$ nor in $\boldsymbol{A} \sqcup x_1$. Especially, it contains an occurrence of x_1. Since M is parameter-reduced, $\boldsymbol{c}(\phi_3) \notin \{-\infty, 0\}$. Hence, by Fact 3.3 for every $k>1$, $\boldsymbol{c}(\psi_2^k\psi_3)=\boldsymbol{c}(\psi_2)^k\boldsymbol{c}(\psi_3)>k$. Also, since M is parameter-reduced, x_1 occurs in $\boldsymbol{c}(\psi_1)$. Hence, by Fact 2.1(2), $\boldsymbol{c}(\psi_1\psi_2^k\psi_3)=\boldsymbol{c}(\psi_1)\boldsymbol{c}(\psi_2)^k\boldsymbol{c}(\psi_3)>k$ and the costs of M cannot be bounded. Therefore, (1) implies (2).

Since implication (3)⇒(1) is again trivial, it only remains to deal with implication (2)⇒(3). Assume ϕ is an accepting computation of A. According to the basic observation in Section 2, $\boldsymbol{c}(\phi)=\bigsqcup\{\boldsymbol{c}(\sigma) \mid \sigma \in \mathrm{ME}(\phi)\}$.

We claim that for every $\sigma \in \mathrm{ME}(\phi)$, some expression σ' of some accepting computation ϕ' exists with $\boldsymbol{c}(\sigma)=\boldsymbol{c}(\sigma')$ such that every node $r \in O(\sigma')$ has length $|r|<n$. Clearly, this claim implies the upper bounds given under (3).

To prove our claim, consider an expression σ' of some accepting computation ϕ' with $\boldsymbol{c}(\sigma)=\boldsymbol{c}(\sigma')$ such that $\#O(\sigma')$ is minimal under all such expressions. For a contradiction, assume $r \geqslant n$ for some node $r \in O(\sigma')$. Then $r=r_1r_2r_3$ for some $r_2 \neq \varepsilon$ such that both $\phi'(r_1)$ and $\phi'(r_1r_2)$ are q-transitions for some state q. We can decompose σ' into $\sigma'=\sigma_1\sigma_2\sigma_3$ where σ_1 is an expression of $\phi'[x_1/r_1]$, σ_2 is an expression of $\phi'/r_1[x_1/r_2]$, and σ_3 is an expression of ϕ'/r_1r_2. Since M has property (A), $\boldsymbol{c}(\sigma_2)=x_1$. Therefore, $\boldsymbol{c}(\sigma')=\boldsymbol{c}(\sigma_1\sigma_3)$. Since $\#O(\sigma_1\sigma_3)<\#O(\sigma')$, this gives a contradiction. □

3.3. The semiring $\boldsymbol{T}$

In this subsection, we consider costs in the semiring $\boldsymbol{T}$. So, let $M=(A, \boldsymbol{c})$ be a $\boldsymbol{T}$-cost automaton where $A=(Q, \Sigma, \delta, Q_F)$. Here, we do not have such a simple characterization for bounded costs as in the two former cases.

The set $M_{\boldsymbol{T}}[X]$ of monomials m over $\boldsymbol{T}$ can be viewed as the set $\boldsymbol{N}^{(1)}[X]$ of polynomials over $\boldsymbol{N}$ of degree at most 1. On $\boldsymbol{N}^{(1)}[X]$ there is a natural partial

ordering $\leqslant$ where for $m_i = h_i + \sum_{j=1}^{k} \varepsilon_{i,j} \cdot x_j$, $m_1 \leqslant m_2$ iff $h_1 \leqslant h_2$ and for all j, $\varepsilon_{1,j} \leqslant \varepsilon_{2,j}$. In this case, $m_1[b_1, \ldots, b_k] \leqslant m_2[b_1, \ldots, b_k]$ for all $b_1, \ldots, b_k \in \mathbb{N}_0$.

A monomial m is *redundant* in polynomial $p \in \boldsymbol{T}[X]$ iff p contains some monomial $m' \neq m$ with $m' \leqslant m$. Redundant monomials m of p can be removed without changing the functional behavior of p. A cost function $\boldsymbol{c}: \delta \to \boldsymbol{T}[X]$ is called *irredundant* iff for every $\tau \in \delta$, $\boldsymbol{c}(\tau)$ contains no redundant monomials. The removal of redundant monomials of $\boldsymbol{c}$ can be done in polynomial time together with the construction of the parameter-reduced automaton M_p corresponding to M. Therefore, we assume w.l.o.g. that M is already parameter-reduced, and that $\boldsymbol{c}$ is irredundant.

Assume ϕ is a computation. Monomial σ of ϕ contains a *costly cycle* iff some $r = r_1 r_2 r_3 \in O(\sigma)$ exists such that

- both $\phi(r_1)$ and $\phi(r_1 r_2)$ are q-transitions for some $q \in Q$;
- $r_1 < r' j \leqslant r_1 r_2$ exists in $O(\sigma)$ such that $\sigma(r') > x_j$.

The main result of this subsection is the following theorem.

Theorem 3.5. *Assume $M = (A, \boldsymbol{c})$ is a parameter-reduced $\boldsymbol{T}$-cost automation such that $\boldsymbol{c}$ is irredundant. Then the following statements are equivalent:*

(1) *M is bounded,*

(2) *for every accepting computation ϕ of A, an expression σ of ϕ exists having no costly cycle,*

(3) $\bigsqcup M \leqslant \begin{cases} H \cdot n & \text{if } k = 1, \\ 2Hk^n & \text{if } k > 1, \end{cases}$

where n is the number of states of A, and H and k are upper bounds of the coefficients and the degrees, respectively, of the polynomials in $\boldsymbol{c}$. It is decidable whether or not a $\boldsymbol{T}$-cost automaton $M = (A, \boldsymbol{c})$ is bounded; the algorithm runs even in polynomial time whenever the degrees of polynomials in $\boldsymbol{c}$ are bounded by 1.

Proof. Assertion (3) trivially implies (1). For deciding boundedness, we can by Corollary 2.4(2), assume w.l.o.g. that $M = (A, \boldsymbol{c})$ is parameter-reduced with irredundant $\boldsymbol{c}$. Provided the upper bounds of (3) hold, we conclude from Corollary 2.4(1) that it can be decided whether or not M is bounded. If this upper bound is polynomial in the size of M, the algorithm even runs in polynomial time.

For a proof that (2) implies (3), let σ be an expression of some accepting computation ϕ without costly cycle. Since $\boldsymbol{c}(\phi) \leqslant \boldsymbol{c}(\sigma)$, it suffices to show that the cost of σ can be bounded appropriately. We claim that some expression σ' of some accepting computation ϕ' exists without costly cycle but with $\boldsymbol{c}(\sigma) = \boldsymbol{c}(\sigma')$ such that every node $r \in O(\sigma')$ has length $|r| < n$. Clearly, this claim implies the upper bounds given under (3). To prove our claim, consider an expression σ' of some accepting computation ϕ' without costly cycle and with $\boldsymbol{c}(\sigma) = \boldsymbol{c}(\sigma')$ such that $\# O(\sigma')$ is minimal under all such expressions. For a contradiction, assume $|r| \geqslant n$ for some node $r \in O(\sigma')$. Then $r = r_1 r_2 r_3$ for some $r_2 \neq \varepsilon$ such that both $\phi'(r_1)$ and $\phi'(r_1 r_2)$ are q-transitions for some state q. We can decompose σ' into $\sigma' = \sigma_1 \sigma_2 \sigma_3$ where σ_1 is an expression of $\phi'[x_1/r_1]$, σ_2 is an expression of $\phi'/r_1[x_1/r_2]$, and σ_3 is an expression of $\phi'/r_1 r_2$. Since σ'

contains no costly cycle, $c(\sigma_2)=x_1$. Therefore, $c(\sigma')=c(\sigma_1\sigma_3)$. Since $\sigma_1\sigma_3$ does not contain a costly cycle either and $\#O(\sigma_1\sigma_3)<\#O(\sigma')$, this gives a contradiction.

It remains to prove that (1) also implies (2). The idea of the proof is the following. Assume (2) does not hold. Then an accepting computation ϕ exists such that every expression of ϕ contains a costly cycle. We pump up all these cycles simultaneously to obtain an accepting computation of arbitrarily high costs. We interrupt the proof in order to introduce some machinery to treat *simultaneous pumping*. Note that some care has to be taken since pumping in one place may introduce new places where pumping has to be performed. To study this formally, we introduce the notion of *residual decompositions*.

Assume $\phi\in\Phi^{(p,\varepsilon)}$. A decomposition f of ϕ is given by proper computations $\phi_0\in\Phi^{(p,q)}$, $\psi\in\Phi^{(q,q\cdots q)}$ and $\phi_1,\ldots,\phi_d\in\Phi^{(q,\varepsilon)}$ such that $\phi=\phi_0\psi[\phi_1,\ldots,\phi_d]$.

Equivalently, the decomposition f of ϕ may be described by a pair $\langle r,R\rangle$ where $r\in O(\phi_0)$ is the leaf in ϕ_0 labeled by x_1 and R is the set of leaves of ψ labeled by variables x_j. r is also called *root* of f.

A node o of ϕ is *factored* by $f=\langle r,R\rangle$ iff $o=rr'o'$ for some $r'\in R$. Assume $k\geqslant 1$. Pumping ϕ up k times w.r.t. f, we obtain computation $\phi^{f,k}=\phi_0\psi^k[\phi_1,\ldots,\phi_d]$.

For ϕ and decomposition $f=\langle r,R\rangle$ of ϕ define $O_{-1}(f)$ as the set of nodes o of ϕ of which r is *not* a prefix; $O_0(f)$ is the set of proper prefixes of elements in R, $O_1(f)$ is the set of nodes in ϕ/r which are incomparable to every node in R; and $O_2(f)$ is the set of nodes o in ϕ/r which have a prefix in R.

Example 3.6. Let $\phi=a(bc,a(a(c,c),c))=\phi_0\psi[\phi_1,\phi_2]$ where $\phi_0=a(bc,x_1)$, $\psi=a(a(x_1,c),x_2)$; $\phi_1=c$ and $\phi_2=c$. Then this decomposition can be written as $f=\langle r,R\rangle$ with $r=2$ and $R=\{1\cdot 1,2\}$.

We have $O_{-1}(f)=\{\varepsilon,1,1\cdot 1\}$; $O_0(f)=\{\varepsilon,1\}$; $O_1(f)=\{1\cdot 2\}$; and $O_2(f)=\{1\cdot 1,2\}$.

Using these definitions, we can partition the nodes of ϕ as follows:

$$O(\phi)=O_{-1}(f)\cup r\cdot O_0(f)\cup r\cdot O_1(f)\cup r\cdot O_2(f).$$

Accordingly, the set of nodes of the kth pump $\phi^{f,k}$ of ϕ is partitioned:

$$O(\phi^{f,k})=O_{-1}(f)\cup r\cdot R^{\leqslant k-1}\cdot O_0(f)\cup r\cdot R^{\leqslant k-1}\cdot O_1(f)\cup r\cdot R^{k-1}\cdot O_2(f)$$

where $R^{\leqslant k-1}=\bigcup\{R^j\mid j\in[0,k-1]\}$.

Thus, every node $o\in O(\phi)$ gives rise to a set $O_o\subseteq O(\phi^{f,k})$, namely,

$$O_0=\begin{cases} o & \text{if } o\in O_{-1}(f),\\ rR^{\leqslant k-1}o_1 & \text{if } o=ro_1 \text{ with } o_1\in O_0(f),\\ rR^{\leqslant k-1}o_1 & \text{if } o=ro_1 \text{ with } o_1\in O_1(f),\\ rR^{k-1}o_1 & \text{if } o=ro_1 \text{ with } o_1\in O_2(f),\end{cases}$$

such that $O(\phi^{f,k})=\bigcup\{O_0\mid o\in O(\phi)\}$.

For our application, we are only interested in nodes o' in O_o which do not have proper prefixes in O_o. Therefore, we define the set $\mathrm{RES}_{f,k}(o)$ of *residuals* of o as the set of *minimal* elements in O_o, i.e.

$$\mathrm{RES}_{f,k}(o)=\begin{cases} o & \text{if } o\in O_{-1}(f),\\ rS^{\leqslant k-1}o_1 & \text{if } o=ro_1 \text{ with } o_1\in O_0(f),\\ rR^{\leqslant k-1}o_1 & \text{if } o=ro_1 \text{ with } o_1\in O_1(f),\\ rR^{k-1}o_1 & \text{if } o=ro_1 \text{ with } o_1\in O_2(f),\end{cases}$$

where $S=\{r'\in R \mid \text{not } o_1<r'\}$.

Note that the definition of $\mathrm{RES}_{f,k}(o)$ differs from O_o only in the second line.

Example (*continued*). Consider the decomposition f of ϕ above. Then

$$\phi^{f,2}=a(bc, a(a(a(a(c, c), c), c), a(a(c, c), c))).$$

For $o=2\in O_0(f)$,

$$\mathrm{RES}_{f,2}(o)=\{2\}.$$

For $o=2\cdot 1\in r\cdot O_0(f)$,

$$\mathrm{RES}_{f,2}(o)=2\cdot\{2\}^{\leqslant 1}\cdot 1=\{2\cdot 1, 2\cdot 2\cdot 1\}$$

and, for $o=2\cdot 1\cdot 2\in r\cdot O_1(f)$,

$$\mathrm{RES}_{f,2}(o)=2\cdot\{1\cdot 1, 2\}^{\leqslant 1}\cdot 1\cdot 2=\{2\cdot 1\cdot 2, 2\cdot 1\cdot 1\cdot 1\cdot 2, 2\cdot 2\cdot 1\cdot 2\}.$$

We have the following result.

Proposition 3.7. (1) *Assume* $o, o'\in O(\phi)$. *If* $o<o'$ *then*

$$\forall o_2\in\mathrm{RES}_{f,k}(o') \quad \exists o_1\in\mathrm{RES}_{f,k}(o): o_1<o_2.$$

(2) *If* $o_1, o_2\in\mathrm{RES}_{f,k}(o)$ *for some* $o\in O(\phi)$ *then* $o_1\neq o_2$ *implies* $o_1\not\approx o_2$.

Besides the decomposition $f=\langle r, R\rangle$ consider a second decomposition $f'=\langle r' R'\rangle$ of ϕ. f' gives rise to a set $\mathrm{RES}_{f,k}(f')$ of *residual decompositions* (in short *residuals*). This set of decompositions of $\phi^{f,k}$ is determined in two steps. First, we compute the sets

$$S_0=\mathrm{RES}_{f,k}(r') \quad \text{and} \quad S_1=\mathrm{RES}_{f,k}(r' R')=\bigcup_{o\in r' R'}\mathrm{RES}_{f,k}(o).$$

The set S_0 gives the set of roots of the residual decompositions. By Proposition 3.7(1), the nodes in S_1 can be partitioned into sets S_w, $w\in S_0$, where S_w contains all nodes from S_1 of which w is a prefix. Define R_w as the set of minimal elements of $\{o_1 \mid wo_1\in S_w\}$. Finally, set $\mathrm{RES}_{f,k}(f')=\{\langle w, R_w\rangle \mid w\in S_0\}$.

We make the following simple observations:

- If $r' \not\approx r$ then $\mathrm{RES}_{f,k}(f') = \{f'\}$.
- If r' is a prefix of r then $\mathrm{RES}_{f,k}\{f'\} = \{\langle r', R''\rangle\}$ for some set of nodes R''.
- $\mathrm{RES}_{f,k}(f) = \{\langle r, R^k\rangle\}$.
- If $r' = ro_1 \in rO_2(f)$ then $\mathrm{RES}_{f,k}(f') = \{\langle o, R'\rangle \mid o \in rR^{k-1}o_1\}$.

Proposition 3.8 follows from the above definition and Proposition 3.7.

Proposition 3.8. *If node o of ϕ is factored by f' then every $\bar{o} \in \mathrm{RES}_{f,k}(o)$ is factored by some $g \in \mathrm{RES}_{f,k}(f')$.*

We return to the investigation of monomial expressions. Assume $f = \langle r, R\rangle$ is the decomposition $\phi = \phi_0\psi[\phi_1, \ldots, \phi_d]$ where ψ is a proper $(q, q \ldots q)$-computation and $\phi_1, \ldots, \phi_d$ are q-computations. Then the set of expressions of $\phi^{f,k}$ consists of all $\sigma = \sigma_0[\sigma']$ where $\sigma_0 \in \mathrm{ME}(\phi_0)$ and σ' is an expression of $\psi^k[\phi_1, \ldots, \phi_d]$ whenever x_1 occurs in σ and ∞ otherwise. $\mathrm{ME}(\psi^k[\phi_1, \ldots, \phi_d])$ is inductively determined as follows.

If $k = 1$ then $\mathrm{ME}(\psi^k[\phi_1, \ldots, \phi_d])$ consists of all expressions $\sigma[\sigma_1, \ldots, \sigma_d]$ where σ is an expression of ψ and σ_j is an expression of ϕ_j provided x_j occurs in σ and ∞ otherwise. If $k > 1$ then $\mathrm{ME}(\psi^k[\phi_1, \ldots, \phi_d])$ consists of all expressions $\sigma[\sigma_1, \ldots, \sigma_d]$ where σ is an expression of ψ and σ_j is an expression of $\psi^{k-1}[\phi_1, \ldots, \phi_k]$ provided x_j occurs in σ and ∞ otherwise.

Note that according to this iterative composition of expressions, an expression of $\phi^{f,k}$ may contain different expressions for different occurrences of the same ϕ_j. Call a set $B \subseteq O(\phi)$ *exhaustive* (for ϕ) if B contains a node of every monomial expression of ϕ. We have the following proposition.

Proposition 3.9. *If a set $B \subseteq O(\phi)$ is exhaustive then $\mathrm{RES}_{f,k}(B)$ is exhaustive as well.*

Proof. Without loss of generality, $f = \langle \varepsilon, R\rangle$. Therefore, let f be the decomposition $\phi = \psi[\phi_1, \ldots, \phi_d]$ where ψ is a proper $(q, q \ldots q)$-computation. We proceed by induction on k. If $k = 1$ then the assertion trivially holds. So assume $k > 1$. Let $\sigma \in \mathrm{ME}(\phi^{f,k})$. Then $\sigma = \sigma'[\sigma_1, \ldots, \sigma_d]$ where $\sigma' \in \mathrm{ME}(\psi)$ and σ_j is an expression of $\psi^{k-1}[\phi_1, \ldots, \phi_k]$ provided x_j occurs in σ (and ∞ otherwise). If B contains some node b of σ' *not* in R then $b \in \mathrm{RES}_{f,k}(b)$ is a node of σ, and the assertion follows. Otherwise, σ' must contain some leaf $r \in R$ labeled with, say x_j. Moreover, by inductive assumption, $\mathrm{RES}_{f,k-1}(B)$ contains some node b' of σ_j. If b' is from $R^{\leqslant k-2}b$ for some $b \in O_1(f)$ or from $R^{k-2}b$ for some $b \in O_2(f)$, rb' is in $\mathrm{RES}_{f,k}(b) \subseteq \mathrm{RES}_{f,k}(B)$, and we are done. Otherwise, $b' \in S^{\leqslant k-2}b$ for some $b \in O_0(f)$ where $S = \{r' \in R \mid \text{not } b < r'\}$. Then either $r \in S$ and $rb' \in \mathrm{RES}_{f,k}(b)$, or $r \notin S$. But then b is a proper prefix of r, and b is a node from $\mathrm{RES}_{f,k}(b)$ in $O(\sigma)$. □

A decomposition $f = \langle r, R\rangle$ of a q-computation ϕ has *cost* at least k iff for every expression σ of ϕ and $r' \in R$ with $rr' \in O(\sigma)$, nodes $r < o_1 j_1 < \cdots < o_k j_k \leqslant rr'$ exist such that for $\kappa = 1, \ldots, k$, $\sigma(o_\kappa) > x_{j_\kappa}$.

Assume $F=\{f_1,\dots,f_m\}$ is a set of decompositions f_μ of ϕ. F is called *complete* for a set $B\subseteq O(\phi)$ iff every node $o\in B$ is factored by some decomposition f_μ in F. If, furthermore, every f_μ has cost at least k, then F is called *k-complete* for B.

Proposition 3.10. (1) *If a node b of an expression σ of ϕ is factored by a decomposition f of ϕ with cost at least k then $c(\sigma)\geqslant k$.*

(2) *If B is exhaustive and ϕ has a set F of decompositions which is k-complete for B then $c(\phi)\geqslant k$.*

Note that Proposition 3.10 no longer holds when M is not parameter-reduced! From Propositions 3.7 and 3.8, we deduce the following proposition.

Proposition 3.11. (1) *If F is complete for B then* $\mathrm{RES}_{f,k}(F)=\bigcup_{f'\in F}\mathrm{RES}_{f,k}(f')$ *is complete for* $\mathrm{RES}_{f,k}(B)$.

(2) *If f' has cost at least h then every decomposition in* $\mathrm{RES}_{f,k}(f')$ *also has cost at least h.*

Proposition 3.12. *Assume ϕ is an accepting computation, $B\subseteq O(\phi)$ is exhaustive, and F is a set of decompositions of ϕ which is 1-complete for B. Then for every $k>1$, an accepting computation ϕ', an exhaustive set $B'\subseteq O(\phi')$ and a set F' of decompositions of ϕ' exist such that F' is k-complete for B'.*

Propositions 3.10 and 3.12 can be applied to obtain a proof of Theorem 3.5.

Proof of Theorem 3.5 (*continued*). Assume statement (2) does not hold. Then some accepting computation ϕ exists such that every expression σ of ϕ has a costly cycle, i.e. $O(\sigma)$ contains some leaf $b_\sigma=r_{\sigma,1}r_{\sigma,2}r_{\sigma,3}$ such that

(1) both $\phi(r_{\sigma,1})$ and $\phi(r_{\sigma,1}r_{\sigma,2})$ are q-transitions for some q;

(2) some node $oj\leqslant r_{\sigma,2}$ exists such that $\sigma(o)>x_j$, and

(3) $r_{\sigma,2}$ is chosen to be of minimal length with this property.

Let $B=\{b_\sigma\mid\sigma\in\mathrm{ME}(\phi)\}$. By assumption, B is exhaustive. A set F of decompositions of ϕ is obtained as follows. First, we collect all nodes $r_{\sigma,1}$ of all costly cycles. Call this set upper set U. Secondly, for every $r\in U$, we collect all lower nodes $r_{\sigma,2}$ where $r_{\sigma,1}=r$. Call this set lower set L_r. We claim that L_r is prefix-free for every $r\in U$. Assume this were not the case. Then expressions σ and σ' exist such that $r_{\sigma,1}=r_{\sigma',1}$ and $r_{\sigma,2}$ is a proper prefix of $r_{\sigma',2}$. By assumption some $r'j\leqslant r_{\sigma,2}$ exists with $\sigma(r')>x_j$. Since $r'j\in O(\sigma')$ also $\sigma'(r')\geqslant x_j$. Since the polynomial $\phi(r')$ contains no redundant monomial, $\sigma'(r')$ must be different from x_j. Hence, we might as well have chosen $r_{\sigma,2}$ as lower node of the costly cycle for σ' – in contradiction to our minimality assumption.

We conclude that L_r is prefix-free. Thus, the set $F=\{\langle r,L_r\rangle\mid r\in U\}$ is a set of decompositions of ϕ. By definition, it is complete for B. Let $\phi=\phi_0\psi[\phi_1,\dots,\phi_m]$ be the decomposition $\langle r,R\rangle$ from F. If a monomial chosen at a node o by some

expression of ϕ is $>x_j$, then by irredundance of c, all monomials chosen for o by any other expression σ' of ϕ containing node oj are greater than x_j. It follows that $\langle r, R\rangle$ has cost at least 1. We conclude that F is even 1-complete for B. Therefore, we can apply Proposition 3.12 to ϕ, B and F to deduce that for every k, there is a computation ϕ', an exhaustive set $B' \subseteq O(\phi')$ and a set F' of decompositions of ϕ' which is k-complete for B. By Proposition 3.10, $c(\phi') \geqslant k$; therefore, M cannot be bounded. □

The rest of this subsection is concerned with a proof of Proposition 3.12.

Proof of Proposition 3.12. Assume the set of decompositions of ϕ is $F = \{f_1, \ldots, f_m\}$ where r_μ is the root of f_μ. Without loss of generality, we assume that

$$\text{if } \mu < \mu' \text{ then } r_\mu \text{ is } \textit{not} \text{ a prefix of } r_{\mu'}. \qquad (*)$$

For $h = 0, \ldots, m$, we inductively define computations ϕ_h, exhaustive sets $B_h \subseteq O(\phi_h)$ and sets F_h of decompositions of ϕ_h.

For $h = 0$, we set $\phi_0 = \phi$, $B_0 = B$ and $F_0 = F$. $\phi_1 = \phi_0^{f_1,k}$, i.e. ϕ_1 is obtained from ϕ_0 by pumping up ϕ k times w.r.t. f_1. Accordingly, $B_1 = \mathrm{RES}_{f_1,k}(B)$ and $F_1 = \mathrm{RES}_{f_1,k}(F_0)$. According to Proposition 3.9, B_1 is exhaustive and F_1 is complete for B_1. By assumption (*), the sets of residual decompositions of $f_2, \ldots, f_m$ consist of just single elements, say $f_{1,2}, \ldots, f_{1,m}$, respectively, where the roots of $f_{1,\mu}$ and f_μ coincide. We can proceed with pumping up ϕ_1 k times w.r.t. $f_{1,2}$ to obtain ϕ_2 where now $B_2 = \mathrm{RES}_{f_{1,2},k}(B_1)$ and $F_2 = \mathrm{RES}_{f_{1,2},k}(F_1)$.

In general, assume ϕ_{h-1}, B_{h-1} and F_{h-1} are already computed and $f_{h-1,h}, \ldots, f_{h-1,m}$ are the decompositions in F_{h-1} with roots $r_h, \ldots, r_m$. Then we define $\phi_h = \phi_{h-1}^{f,k}$, $B_h = \mathrm{RES}_{f,k}(B_{h-1})$ and $F_h = \mathrm{RES}_{f,k}(F_{h-1})$ where $f = f_{h-1,h}$.

We claim that B_m is exhaustive and F_m is k-complete for B_m. For a proof of this claim call a set F of decompositions of ϕ k-complete for set B up to $F' \subseteq F$ iff F is complete for B and the following hold:

(1) every decomposition $f \in F \setminus F'$ has cost at least k,

(2) every decomposition $f \in F'$ has cost at least 1.

We claim that for $h = 0, \ldots, m$,

- B_h is exhaustive,
- F_h is k-complete for B up to $\{f_{h,h+1}, \ldots, f_{h,m}\}$.

Proposition 3.12 is the special instance of this statement where $h = m$. The first assertion follows by induction on h from Proposition 3.9. The second claim is proved in three steps: For every $h \in \{0, \ldots, m\}$,

- F_h is complete for B_h,
- every decomposition in F_h has cost at least 1,
- every decomposition in $F_h \setminus \{f_{h,h+1}, \ldots, f_{h,m}\}$ has cost at least k.

By assumption, $F_0 = F$ is 1-complete for $B_0 = B$. Therefore, the first two statements follow from Proposition 3.11(1) and (2) by induction on h.

The third statement is also proved by induction on h. For $h = 0$, it trivially holds. Assume the assertion holds for $h-1$. Then the inductive step "$h-1 \to h$" follows from

Proposition 3.11(1) and the fact that if decomposition f has cost at least 1 then the residual decomposition in $\mathrm{RES}_{f,k}(f)$ has cost at least k. □

3.4. The semiring $\boldsymbol{F}$

Finally, we consider costs in the semiring $\boldsymbol{F}$. As for semirings $\boldsymbol{N}$ or $\boldsymbol{A}$, we would like to concentrate on the form of costs occurring in strong components of the trace graph. However, now there are several possibilities how the cardinality of costs may increase. We have, in this connection, the following fact.

Fact 3.13. *Assume* $p \in \boldsymbol{F}[x]$ *and* $b \notin \{\emptyset, \{0\}\}$. *Then* $\#(p^h[b]) > h$ *for every* $h \in \mathbb{N}$ *or* $\#(p^h[b]) = \#(p[b])$ *and one of the following statements hold:*

- $p = a$ *or* $p = a \cup x$ *for some* $a \in \boldsymbol{F}$,
- $p = a + j \cdot x$ *where* $\#a = 1$ *and* $j = 1$,
- $p = a + j \cdot x$ *where* $\#a = 1$, $j > 1$ *and* $\#b = 1$.

Moreover, we need the following simple observation about polynomials from $\boldsymbol{N}^{(1)}[X_k]$.

Fact 3.14. *Let* m_1, m_2 *be polynomials in* $\boldsymbol{N}^{(1)}[X_k]$ *and* $a_{1,j}, a_{2,j} \in \boldsymbol{N}$ *such that* $a_{1,j} \neq a_{2,j}$ *for* $j \in [1, k]$. *If* $m_1[a_{\mu(1),1}, \ldots, a_{\mu(k),k}] = m_2[a_{\mu(1),1}, \ldots, a_{\mu(k),k}]$ *for all mappings* $\mu : [1, k] \to \{1, 2\}$ *then* $m_1 = m_2$.

Assume $M = (A, \boldsymbol{c})$ is an $\boldsymbol{F}$-cost automaton with $A = (Q, \Sigma, \delta, Q_F)$. For $q \in Q$, define M_q as the $\boldsymbol{F}$-cost automaton obtained from M by replacing the set Q_F of accepting states with $\{q\}$. M has property (F) iff

(F) every strong component Q' of $G(A)$ is of one of the following three types:

Type I: $Q' \subseteq \{q \in Q \mid \#M_q \leq 1\}$.

Type II: For every edge $(q, \langle \tau, j \rangle, q_j)$ in Q' with $\tau = (q, a, q_1 \ldots q_k)$, $\boldsymbol{c}(\tau) = x_j \cup p$ for some polynomial p in which x_j does not occur and where $\#(\boldsymbol{c}(A_{q_i})) < \infty$ for every x_i occurring in p.

Type III: for every edge $(q, \langle \tau, j \rangle, q_j)$ in Q' with $\tau = (q, a, q_1 \ldots q_k)$, $\boldsymbol{c}(\tau) = x_j + p$, where $p = H + \sum_{i=1}^{k} \varepsilon_i \cdot x_i$ with $\#H = 1$, $\varepsilon_j = 0$ and $\#M_{q_i} \leq 1$ for every x_i occurring in p.

Property (F) collects situations considered "harmless" w.r.t. to pumping. Consider state q of strong component Q' together with a proper (q, q)-computation $\phi_1 \neq x_1$ and a q-computation ϕ_2. Whenever Q' is of type I, clearly the cardinality of $\boldsymbol{c}(\phi_1^k \phi_2)$ is at most one for every k. If Q' is of type II, property (F) ensures that for every k, $\boldsymbol{c}(\phi_1^k \phi_2)$ is contained in $\boldsymbol{c}(\phi_2) \cup B$ for some finite set B. Finally, if Q' is of type III, then (at least provided M is parameter-reduced) the values of $\boldsymbol{c}(\phi_1^k \phi_2)$ may be different but all have the same cardinality.

Opposed to properties (N) and (A), it is not at all clear that property (F) can be decided in polynomial time. Also, property (F) only characterizes boundedness of M provided M is in a special normal form.

A state $q \in Q$ is called *constant* iff $c(\phi) = c(\phi')$ for every two q-computations ϕ and ϕ'. Let $\mathrm{Const}(A, c)$ denote the set of all constant states, and define a function $c : \mathrm{Const}(A, c) \to \boldsymbol{F}$ by $c(q) = B$ iff $c(\phi) = B$ for some q-computation ϕ. A state $q \in Q$ is called 1-*constant* iff q is constant and $\# c(q) \leqslant 1$. The set of all 1-constant states of A is denoted by $\mathrm{Const}_1(A, c)$. $M = (A, c)$ is called 1-*strongly reduced* iff M is parameter-reduced and $\mathrm{Const}_1(M) \cap [Q \setminus Q_F] \subseteq U_\emptyset(M)$. It turns out that at least the set of 1-constant states can be computed in polynomial time. This gives rise to Proposition 3.15 which guarantees that we may assume w.l.o.g. always that the cost automata in question are 1-strongly reduced. The subsequent two propositions provide the two crucial subroutines necessary to decide property (F), namely, an algorithm deciding whether or not for a given $\boldsymbol{F}$-cost automaton $M = (A, c)$, $\# c(A) < \infty$ (Proposition 3.16), and an algorithm deciding whether or not for a given $\boldsymbol{F}$-cost automaton M, $\# M \leqslant 1$ (Proposition 3.17).

Proposition 3.15. *For every parameter-reduced $\boldsymbol{F}$-cost automaton $M = (A, c)$, a 1-strongly reduced $\boldsymbol{F}$-cost automaton $M_s = (A_s, c_s)$ can be constructed in polynomial time such that $L(A) = L(A_s)$ and $c(A) = c_s(A_s)$.*

Proof. First, we compute the 1-constant states. The algorithm doing so is a special instance of grammar flow analysis as introduced in [6].

Let $A = (Q, \Sigma, \delta, Q_F)$. Without loss of generality, assume M is already parameter-reduced and $f \in Q_F$ implies $f \neq q_j$ for every $(q, a, q_1, \ldots, q_k) \in \delta$. Assume $\boldsymbol{R}$ is the ordered semiring with carrier $\{\bot, \mathrm{ERR}, \emptyset\} \cup \{\{i\} \mid i \in \mathbb{N}_0\}$ where the ordering is given by $\bot < \emptyset < \mathrm{ERR}$ and $\bot < \{i\} < \mathrm{ERR}$ for $i \in \mathbb{N}_0$; addition is $\cup$ defined by $\mathrm{ERR} \cup x = x \cup \mathrm{ERR} = \mathrm{ERR}$, $\bot \cup x = x \cup \bot = \bot$ provided $x \neq \mathrm{ERR}$, $\emptyset \cup x = x \cup \emptyset = x$, $x \cup x = x$, and $\{x\} \cup \{y\} = \mathrm{ERR}$ provided $x \neq y$. Multiplication is defined by $\emptyset + x = x + \emptyset = \emptyset$, $\bot + x = x + \bot = \bot$ whenever $x \neq \emptyset$, $\mathrm{ERR} + x = x + \mathrm{ERR} = \mathrm{ERR}$ whenever $x \notin \{\emptyset, \bot\}$, and $\{x\} + \{y\} = \{x + y\}$. Note that both addition and multiplication of $\boldsymbol{R}$ are indeed monotone (and hence even continuous). There is a semiring morphism $\tilde{(\_)} : \boldsymbol{F} \to \boldsymbol{R}$ defined by $\tilde{A} = A$ if $\# A \leqslant 1$ and $\tilde{A} = \mathrm{ERR}$ otherwise.

Consider the trace graph $G(A) = (V, E)$ and define a map $\mathrm{OUT} : V \to \boldsymbol{R}$ as the least upper bound of mappings OUT_i, $i \in \mathbb{N}_0$, where

$$\mathrm{OUT}_0(q) = \bot;$$

and

$$\mathrm{OUT}_i(q) = \bigsqcup_{\tau = (q, a, q_1 \ldots q_k) \in \delta} \tilde{c}(\tau)[\mathrm{OUT}_{i-1}(q_1), \ldots, \mathrm{OUT}_{i-1}(q_k)] \text{ for } i > 0.$$

We have:

(1) $q \in Q$ is 1-constant iff $\mathrm{OUT}(q) \neq \mathrm{ERR}$.

(2) $\mathrm{OUT}(q) = c(q)$ whenever $q \in \mathrm{Const}_1(M)$.

The map OUT(_) can be computed from M by a RAM in polynomial time which can perform additions and multiplications in $\mathbb{N}_0$ in constant time. The values of OUT(_) different from ERR or $\emptyset$ are bounded by $2 \cdot H \cdot k^n$ where k is the maximal degree of a polynomial $c(\tau)$ and H is the maximal occurring coefficient in $\mathbb{N}_0$. Hence, the lengths of the values are polynomial in the input size. Therefore, the algorithm when implemented on a Turing machine also runs in polynomial time.

Now, define $A_s = A$ and c_s as follows. Assume $\tau = (q, a, q_1 \dots q_k) \in \delta$. If $q \in Q_F \cap \mathrm{Const}_1(M)$ then $c_s(\tau) = c(q)$. If $q \in \mathrm{Const}_1(M) \setminus Q_F$ then $c_s(q) = \emptyset$. Otherwise, $c_s(\tau) = c(\tau)[s_1, \dots, s_k]$ where

$$s_j = \begin{cases} c(q_j) & \text{if } q_j \in \mathrm{Const}_1(M) \\ x_j & \text{otherwise.} \end{cases} \quad \square$$

Proposition 3.16. *It can be decided in polynomial time for an* $\boldsymbol{F}$*-cost automaton* $M = (A, c)$ *whether or not* $\# c(A)) < \infty$.

Proof. There is a semiring morphism $\pi : \boldsymbol{F} \to \boldsymbol{A}$ given by $\pi(B) = \bigsqcup_{b \in B} b$. Define $M_\pi = (A, c_\pi)$ as the $\boldsymbol{A}$-cost automaton where $c_\pi = \pi c$. We have: $\# (c(A)) < \infty$ iff $\# \bigcup \{c(\phi) \mid \phi$ accepting$\} < \infty$ iff $\bigsqcup \{\pi c(\phi) \mid \phi$ accepting$\} < \infty$ iff $\bigsqcup M_\pi < \infty$. Since the latter can be decided in polynomial time by Theorem 3.4, we are done. $\square$

Proposition 3.17. *It can be decided in polynomial time for a given parameter-reduced* $\boldsymbol{F}$*-cost automaton* M *whether or not* $\# M \leqslant 1$.

Putting Propositions 3.16 and 3.17 together, we immediately find that property (F) can indeed be decided in polynomial time.

Corollary 3.18. *It can be decided in polynomial time whether or not a* 1*-strongly reduced* $\boldsymbol{F}$*-cost automaton* M *has property* (F).

Proof of Proposition 3.17. The method we apply here is inspired by techniques used in [9] when dealing with single-valued transducers. Especially, we learn from [9] that we may find a simple syntactical characterization of $\# M \leqslant 1$ provided M is in a special normal form, namely, it is 1-strongly reduced.

Assume $M = (A, c)$ is parameter-reduced. M has property (U1) iff

(U1) For every $\tau = (q, a, q_1 \dots q_k) \in \delta$, $c(\tau) = \emptyset$ or $c(\tau) = \{h\} + \sum_{j=1}^{k} \varepsilon_j \cdot x_j$ for some $h \in \mathbb{N}_0$ and $\varepsilon_j \in \mathbb{N}_0$.

Clearly, it can be decided in polynomial time whether or not M has property (U1). We claim:

Assume M *is* 1*-strongly reduced. Then* $\# M \leqslant 1$ *iff* M *has property* (U1). (*)

By Proposition 3.15, we may assume w.l.o.g. that M is 1-strongly reduced. Therefore, claim (*) implies Proposition 3.17. It remains to prove claim (*). If M has property (U1) then clearly $\# M \leqslant 1$. For the converse implication, assume $\# M \leqslant 1$ and all coefficients occurring in the image of δ have cardinality at most 1. Thus (by ignoring set brackets) we can view the occurring monomials as polynomials from $N^{(1)}[X]$. For a contradiction, assume some $\tau = (q, a, q_1 \dots q_k) \in \delta$ exists such that $c(\tau) = m_1 \cup \cdots \cup m_r$ for monomials m_i where, e.g., $m_1 \neq m_2$. Consider a variable x_j occurring either in m_1 or m_2. Since M is 1-strongly reduced and $\# M \leqslant 1$ by assumption there are q_j-computations $\phi_{1,j}$ and $\phi_{2,j}$ such that $c(\phi_{1,j}) \neq c(\phi_{2,j})$. Thus, Fact 3.14 implies that some computation $\phi = \tau(\phi_1, \dots, \phi_k)$ exists with $m_1[c(\phi_1), \dots, c(\phi_k)] \neq m_2[c(\phi_1), \dots, c(\phi_k)]$ and hence, $\# c(\phi) > 1$. By parameter-reducedness of M and Fact 2.1(2), this contradicts $\# M \leqslant 1$. We conclude that M has property (U1) whenever $\# M \leqslant 1$.

We are now ready to state and prove the main theorem of this subsection.

Theorem 3.19. *For a 1-strongly reduced* **F***-cost automaton $M = (A, c)$ the following three statements are equivalent:*

(1) *M is bounded,*

(2) *M has property* (F),

(3) $\# M \leqslant \begin{cases} [(L+1) \cdot n \cdot (Hn+1)]^n & \text{if } k = 1, \\ [(L+1)^k \cdot 2H \cdot n \cdot k^n]^{k^n} & \text{if } k > 1, \end{cases}$

where n is the number of states of A, L is the rank of the input signature, H and k are upper bounds for the elements of the coefficients and degrees of polynomials occurring in c, respectively.

It can be decided in polynomial time whether or not a given **F***-cost automaton M is bounded.*

Proof. Provided the characterization of boundedness by property (F) holds, the decidability result follows from Proposition 3.15 and Corollary 3.18. It, therefore, suffices to show equivalence of statements (1)–(3).

(3) trivially implies (1). To prove that (1) implies (2), we build on Fact 3.13. If for every state q in every strong component of A, $M_q \leqslant 1$, then clearly $\# M < \infty$. Therefore, assume that Q' is a strong component of A where for some state $q \in Q'$ (and, hence, by parameter-reducedness of M, for every state of Q'), $\# M_q > 1$. Consider an edge $(q, \langle \tau, j \rangle, q_j)$ in Q' where $\tau = (q, a, q_1 \dots q_k) \in \delta$. We claim that $c(\tau)$ is of one of the forms occurring in strong components of Types II or III.

By parameter-reducedness of M, computations ϕ_1, ϕ_3 exists where ϕ_1 is a proper (f, q)-computation for some $f \in Q_F$ such that $c(\phi_1)$ depends on x_1, and ϕ_3 is a q-computation with $\# c(\phi_3) > 1$. Thus, especially, $c(\phi_3) \notin \{\emptyset, \{0\}\}$. By parameter-reducedness of M, a proper (q, q)-computation $\phi_2 = \tau(\psi_1, \dots, \psi_k)$ exists where j is prefix of the occurrence of x_1 in ϕ_2 and $c(\psi_j)$ depends on x_1. We prove our claim by contradiction. We distinguish several cases.

Case 1: $c(\tau)=m\cup p$ for some polynomial p and a monomial

$$m=H+\sum_{i=1}^{k}\varepsilon_i\cdot x_i.$$

Case 1.1: $\#H>1$ or m depends on some x_i, $i\neq j$, where $\#M_{q_i}>1$. In the latter case, we may choose ψ_i in such a way that $\#c(\psi_i)>1$. It follows that $c(\phi_2)=m'\cup q$ for some monomial $m'=H'+\varepsilon\cdot x_1$ where $\#H'>1$. If $\#H'>1$ then for every $h\geqslant 1$, $\#[h\cdot H']>h$. Therefore,

$$\#[c(\phi_2^h\phi_3)]=\#[c(\phi_2)^h c(\phi_3)]\geqslant\#[m'^h c(\phi_3)]\geqslant\#[h\cdot H']>h.$$

We conclude by Fact 2.1 that for every $h\geqslant 1$, $\#c(\phi_1\phi_2^h\phi_3)>h$, which contradicts the boundedness of M.

Case 1.2: $\varepsilon_j>1$. Then $c(\phi_2)=m'\cup q$ with $m'=H'+\varepsilon x_1$ where $\varepsilon\geqslant\varepsilon_j>1$. Therefore,

$$\#[c(\phi_2^h\phi_3)]=\#[c(\phi_2)^h c(\phi_3)]\geqslant\#[m'^h c(\phi_3)]\geqslant\#[\varepsilon^h\cdot c(\phi_3)]>\varepsilon^h,$$

since $\#c(\phi_3)>1$. Again we conclude that M cannot be bounded.

Case 2: Case 1 does not hold for any edge in Q' but $c(\tau)=m_1\cup m_2\cup p$ for distinct $m_v=H_v+\sum_{i=1}^{k}\varepsilon_{v,i}\cdot x_i$ where $\#H_v=1$ and $\varepsilon_{v,j}=1$ for $v=1,2$, and $\#M_{q_i}=1$ for every $i\neq j$ with $\varepsilon_{v,i}\neq 0$. Let $J=\{i\neq j\mid m_1\cup m_2$ depends on $x_i\}$. Since M is 1-strongly reduced, no state q_i, $i\in J$, can be 1-constant. Without loss of generality, we can assume that we can find for every $i\in J$, q_i-computations $\psi_{i,1}$ and $\psi_{i,2}$ such that $c(\psi_{i,1})\neq c(\psi_{i,2})$ with $\#c(\psi_{i,v})=1$. By ignoring set brackets we can view the monomials m_v as polynomials from $N^{(1)}[X]$. Therefore, we can apply Fact 3.14. We deduce that ψ_i, $i\neq j$, can be chosen such that $c(\tau(\psi_1,\ldots,\psi_{j-1},x_1,\psi_{j+1},\ldots,\psi_k))$ contains some monomial $H+x_1$, $v=1,2$, where $n_1,n_2\in H$ and $n_1\neq n_2$. Since case 1 does not hold, we can choose ψ_j such that $c(\phi_2)$ contains a monomial $H'+x_1$ with $H'\neq\emptyset$. Hence, for $\phi=\tau(\psi_1,\ldots,\psi_k)$,

$$c(\phi_2)=H+H'+x_1\cup q'$$

for some polynomial q' where $\#[H+H']\geqslant\#H\geqslant 2$. Now the same argumentation as for case 1.1 shows that M cannot be bounded: contradiction.

Case 3: Neither case 1 nor case 2 holds for any edge in Q' but $c(\tau)=m\cup p$ for some polynomial $p\neq\emptyset$ independent of x_j where $m=\{n\}+x_j$ with $n\neq 0$. If neither case 1 nor case 2 holds for any edge in Q', then $c(\phi_2)=\{n'\}+x_1\cup B$ for some $B\neq\emptyset$ with $n'\neq 0$. Hence for $h>0$,

$$c(\phi_2)^h=\bigcup_{i=0}^{h-1}\{i\cdot n'\}+B\cup\{h\cdot n'\}+x_1.$$

We conclude that $\#[c(\phi_2^h\phi_3)]\geqslant h$, giving a contradiction with $\#M<\infty$.

Case 4: Neither case 1 nor case 2 nor case 3 holds for some edge in Q' but $c(\tau)=x_1\cup p$ for some polynomial $p\neq\emptyset$ which does not depend on x_j but on some x_i with $\#c(A_{q_i})=\infty$. Then we can find a sequence $\psi_1,\ldots,\psi_r,\ldots$ of q_i-computations such that $c(\psi_r)$ contains some $x\geqslant r$. By parameter-reducedness of M, a proper

$(q, q_i q)$-computation ϕ_2 exists such that $c(\phi_2)=p_1 \cup p_2$ where x_1 only occurs in p_1, and x_2 only occurs in p_2.

For every $h>0$, we construct an accepting computation $\phi^{(h)}$ by iterating ϕ_2 h times and inserting computations ψ_i into the different instances of ϕ_2. Precisely, define $\phi^{(h)}=\phi_1 \tilde{\phi}^{(h)}$ where $\tilde{\phi}^{(0)}=\phi_3$ and for $i>0$, $\tilde{\phi}^{(i)}=\phi_2[\psi_r, \tilde{\phi}^{(i-1)}]$ where r is larger than the maximal element in $p_2 c(\tilde{\phi}^{(i-1)})$. Since p_1 depends on $x_1, p_1 c(\psi_r)$ also contains an element $x \geqslant r$ which therefore is not in $p_2(\tilde{\phi}^{(i-1)})$. Hence, we have $\# c(\tilde{\phi}^{(0)})>0$, and for $i>0$,

$$\begin{aligned}
\# c(\tilde{\phi}^{(i)}) &= \# c(\phi_2[\psi_r, \tilde{\phi}^{(i-1)}]) \\
&= \#[p_1(c(\phi_r)) \cup p_2(c(\tilde{\phi}^{(i-1)}))] \\
&\geqslant 1+\# p_2(c(\tilde{\phi}^{(i-1)})) \\
&\geqslant 1+\# c(\tilde{\phi}^{(i-1)})>1+(i-1)=i.
\end{aligned}$$

Since x_1 occurs in $c(\phi_1)$, we conclude that for all $h>0$,

$$\# c(\phi^{(h)})=\#[c(\phi_1) c(\tilde{\phi}^{(h)})]>h$$

in contradiction to $\# M<\infty$.

This finishes the proof of our claim. It remains to prove that cost polynomials of both forms $x_j \cup p$ and $H+\sum_i \varepsilon_i \cdot x_i$ cannot occur within the same strong component. So, assume we are given edges $(q, \langle \tau, j\rangle, q_j)$ and $(p, \langle \tau', j'\rangle, p_{j'})$ in Q' where $\tau=(q, a, q_1 \ldots q_k)$, $\tau'=(p, b, p_1 \ldots p_l) \in \delta$ with $c(\tau)=x_j \cup p$ such that $p \neq \emptyset$ does not depend on x_j and $c(\tau')=\{n\}+\sum_i \varepsilon_i \cdot x_i$ such that $\varepsilon_j=1$. Since M is parameter-reduced, computations $\phi_1, \phi_2, \phi_3, \phi_4$ of A exist such that ϕ_1 is a proper (f, q)-computation for some $f \in Q_F$, ϕ_2 is a proper (q, p)-computation with $\phi_2(\varepsilon)=\tau$, ϕ_3 is a proper (p, q)-computation with $\phi_3(\varepsilon)=\tau'$, and ϕ_4 is a q-computation with $c(\phi_4) \notin \{\emptyset, \{0\}\}$.

Without loss of generality we may assume that $c(\phi_2)=x_1 \cup B$ for some $B \neq \emptyset$, and $c(\phi_3)$ contains a monomial $m=H+x_1$ for some $H \notin \{\emptyset, \{0\}\}$. It follows that $c(\phi_2 \phi_3)=H'+x_1 \cup B'$ for some $B' \neq \emptyset$. But then analogous to cases 1 and 3, we can deduce that $\# M$ cannot be finite in contradiction to the boundedness of M. Thus, statement (1) of the theorem implies statement (2).

It remains to prove that (2) implies the upper bound for $\# M$ given by statement (3). First observe that $\pi: \boldsymbol{F} \to \boldsymbol{A}$ given by $\pi(B)=\bigsqcup_{b \in B} b$ such that $\#(c(A_q))<\infty$, implies that $\# M_q \leqslant Hn+1$ in case $k=1$ and $\# M_q \leqslant 2 \cdot H \cdot k^n$ otherwise.

Secondly, observe that $\#(A \cup B) \leqslant \# A+\# B$ and $\#(A+B) \leqslant \# A \cdot \# B$. These observations allow similar calculations as in the proof of Theorem 3.2 which gives the desired result. □

4. Multidimensional cost automata

Although cost functions as considered so far may suffice for a lot of interesting cases (see e.g. the examples in the introduction), MS-evaluations in general give rise to

multidimensional cost functions [1]. Therefore, in this section, we extend our methods to the multidimensional case. We succeed in proving results concerning boundedness at least in case of the semirings $\boldsymbol{N}$ and $\boldsymbol{A}$.

For the following, we fix a dimension $d \in \mathbb{N}$. Let now X denote the *doubly indexed* set of variables $\{x_{i,j} \mid i \in \mathbb{N},\ j \in [1,d]\}$, and $X_k = \{x_{i,j} \mid i \in [1,k],\ j \in [1,d]\}$ for every $k \in \mathbb{N}$. Assume $A = (Q, \Sigma, \delta, Q_F)$ is an FTA and $\boldsymbol{R}$ denotes a semiring. A *d-dimensional $\boldsymbol{R}$-cost function* for A is a mapping $c: \delta \to \boldsymbol{R}[X]^d$ where $c(\tau) \in \boldsymbol{R}[X_k]^d$ provided $\tau = (q, a, q_1 \dots q_k)$.

$c(\_)$ is extended to computations ϕ in the natural way. If $\phi = x_j$ then $c(\phi)_\mu = x_{j,\mu}$ for all $\mu \in [1,d]$. (As usual, we write the μth component of $c(\_)$ as $c(\_)_\mu$.) If $\phi = \tau(\phi_1, \dots, \phi_m)$ for some $\tau \in \delta$ then $c(\phi) = c(\tau)[c(\phi_1), \dots, c(\phi_m)]$ where substitution is extended to tuples in the natural way.

An *$\boldsymbol{R}$-cost automaton* M of dimension d is a pair $M = (A, c)$ where A is the FTA underlying M, and $c: \delta \to \boldsymbol{R}[X]^d$ is a d-dimensional $\boldsymbol{R}$-cost function for A.

The *size* of M again consists of the size of A together with the space to represent c. Hence, we define $|M| = |A| + \sum_{\tau \in \delta} \sum_{j=1}^{d} |c(\tau)_j|$.

In case $\boldsymbol{R} \in \{\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{F}\}$, we may define the set of $\boldsymbol{R}$-costs of M, $c(A)$, by

$$c(A) = \{c(\phi)_1 \mid \phi \text{ accepting computation of } A\}.$$

Accordingly, for $\boldsymbol{R} \in \{\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{T}\}$, $\bigsqcup M = \bigsqcup c(A)$ is the least upper bound for $\boldsymbol{R}$-costs of M. The notions of E-reducedness and E-parameter-reducedness can be extended to d-dimensional cost automata in a straightforward way. Especially, instead of sets of states $U_r(M)$, we need sets of states $U_{\langle r_1, \dots, r_d \rangle}(M)$ for every d-tuple $\langle r_1, \dots, r_d \rangle \in (E \cup \{\bot\})^d$ where symbol $\bot$ denotes a cost $\notin E$.

Theorem 4.1. *Assume $\boldsymbol{R}$ is a semiring and $E \subseteq \boldsymbol{R}$ is faithful via $H: \boldsymbol{R} \to \boldsymbol{R}'$ with $0 \in E$. Then the following holds:*

(1) *For every d-dimensional $\boldsymbol{R}$-cost automaton $M = (A, c)$ with n states there is an E-reduced $\boldsymbol{R}$-cost automaton $M_r = (A_r, c_r)$ such that*

- *$L(A) = L(A_r)$ and $c(A) = c_r(A_r)$;*
- *A_E has at most $(\#\boldsymbol{R}')^d \cdot n$ states and $|M_r| \leqslant |M| \cdot (\#\boldsymbol{R}')^{dL}$;*
- *M_r can be constructed by a RAM in polynomial time.*

(2) *For every E-reduced $\boldsymbol{R}$-cost automaton $M = (A, c)$ with n states there is an E-parameter-reduced $\boldsymbol{R}$-cost automaton $M_p = (A_p, c_p)$ such that*

- *$L(A) = L(A_p)$ and $c(A) = c_p(A_p)$;*
- *A_p has at most $2^d \cdot n$ states and $|M_p| \leqslant |M| \cdot 2^d$;*
- *M_p can be constructed by a RAM in polynomial time.*

From Theorem 4.1 we deduce the following corollary.

Corollary 4.2. (1) *For $\boldsymbol{R} \in \{\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{T}\}$, it can be decided for a given d-dimensional $\boldsymbol{R}$-cost automaton $M = (A, c)$ and $m \in \mathbb{N}$ in time polynomial in $|M|$ and m, whether or not $\bigsqcup M < m$.*

For fixed $m \in \mathbb{N}$, it can be decided for a given d-dimensional $\boldsymbol{F}$-cost automaton $M=(A, \boldsymbol{c})$ in polynomial time whether or not $\boldsymbol{c}(A) \subseteq 2^{[0, m-1]}$.

(2) *Let $\boldsymbol{R} \in \{\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{F}\}$. For every $\boldsymbol{R}$-cost automaton $M=(A, \boldsymbol{c})$, a parameter-reduced $\boldsymbol{R}$-cost automaton $M_p=(A_p, \boldsymbol{c}_p)$ can be constructed in polynomial time with $L(A)=L(A_p)$ and $\boldsymbol{c}(A)=\boldsymbol{c}_p(A_p)$.*

4.1. The semiring $\boldsymbol{N}$

In this subsection we consider again costs in the semiring $\boldsymbol{N}$. So, let $M=(A, \boldsymbol{c})$ be an $\boldsymbol{N}$-cost automaton of dimension d where $A=(Q, \Sigma, \delta, Q_F)$. Without loss of generality we may assume M is parameter-reduced.

Fact 4.3. *Assume $p_1, \ldots, p_d$ are nonconstant polynomials in $\boldsymbol{N}[X_{1,d}]$ such that some polynomial p_j exists that depends on $x_{1,j'}$ for every $j' \in [1, d]$. Let $b=\langle b_1, \ldots, b_d \rangle \in \boldsymbol{N}^d$ where $b_j > 1$ for all j. Then either* (1) *or* (2) *is true*:

(1) *For every j, $p_j = x_{1,j'}$ for some $j' \in [1, d]$.*

(2) *There exists some $j \in [1, d]$ such that for every $k \in \boldsymbol{N}$, $p^n[b]_j > k$ for some $n \in \mathbb{N}$.*

Proof. By Fact 2.1 and induction on k, we find that for every $j \in [1, d]$, $p_j[p^{k-1}[b]] > 1$. Moreover, $p_j[p^{k-1}[b]] > p_{j'}[p^{k-2}[b]]$ provided at least one of the following three properties holds:

(*1) Some $j_1 \neq j'$ exists such that x_{1,j_1} occurs in p_j as well.

(*2) p_j contains two monomials in which $x_{1,j'}$ occurs.

(*3) $p_j = h_0 + h_1 \cdot x_{1,j'}^{\varepsilon}$ such that either $h_0 > 0$, $h_1 > 1$ or $\varepsilon > 1$.

If no polynomial p_j has one of the properties $(*i)$ for some j' then Fact 4.3 holds. Therefore, assume that polynomial p_j exists such that at least one of the properties $(*i)$ holds for some j'. Let G denote the directed graph $G=(V, E)$ with $V=[1, d]$ and $E=\{(j', j) \mid p_j \text{ depends on } x_{1,j'}\}$ as set of edges.

Since no polynomial is constant and every $x_{1,j'}$ occurs in some $p_{j'}$ every node j' is source of an edge (j', j). It follows that a strong component of G can be reached from every node. We mark all edges (j', j) in G where for p_j and j' one of properties (*1), (*2) or (*3) holds. Assume every strong component of G has no marked edge. It follows that every node in a strong component has indegree 1. Hence, G consists of a disjoint set cycles which means that G has no marked edge at all in contradiction to our assumption. Hence, G contains some cycle with a marked edge. This cycle has some length $r \in \mathbb{N}$. Assume j is a node on this cycle. It follows that $p^{(k+1) \cdot r}[b]_j > p^{k \cdot r}[b]_j$ for all k. Hence, $p^n[b]_j > k$ for $n = k \cdot r$. □

The d-dimensional $\boldsymbol{N}$-cost automaton $M=(A, \boldsymbol{c})$ has property (N) iff

(N) For every edge $(q, \langle \tau, j \rangle, q')$ of a strong component of the trace graph $G(A)$ and $\mu \in [1, d]$, $\boldsymbol{c}(\tau)_\mu = 0$ or $\boldsymbol{c}(\tau)_\mu = x_{j,\nu}$ for some $\nu \in [1, d]$.

Clearly, it can be decided in polynomial time whether or not M has property (N). Now, we have the following theorem.

Theorem 4.4. *Assume $M=(A, \boldsymbol{c})$ is a parameter-reduced $\mathbf{N}$-cost automaton of dimension d. The following three statements are equivalent:*

(1) $\bigsqcup M<\infty$,

(2) *M has property* (N),

(3) $\bigsqcup M \leqslant \begin{cases}[(L+1)\cdot H]^n & \text{if } k=1,\\ [(L+1)^k\cdot H]^{k^n} & \text{if } k>1,\end{cases}$

where n is the number of states of A, L is the rank of Σ, H is an upper bound to the coefficients of polynomials occurring in $\boldsymbol{c}(\delta)$ and k is an upper bound to the degree of these polynomials.

It can be decided in polynomial time whether or not $\bigsqcup M$ is finite.

Proof. Assume M does not have property (N). Then there is an accepting computation $\phi=\psi_1\psi_2\psi_3$ for proper (f, q)-computation ψ_1, proper (q, q)-computation ψ_2 and q-computation ψ_3 such that $\boldsymbol{c}(\psi_2)_j\notin\{0\}\cup X$ for some j. Assume $S\subseteq[1, d]$ is the set of all j such that $\boldsymbol{c}(\psi_2)_j\neq 0$. Since M is parameter-reduced, all polynomials $\boldsymbol{c}(\psi_2)_j$, $j\in S$, are nonconstant, and some $j\in S$ exists for every $j'\in S$, such that $\boldsymbol{c}(\psi_2)_j$ depends on $x_{1,j'}$. Therefore, we can apply Fact 4.3 to $\langle \boldsymbol{c}(\psi)_j\rangle_{j\in S}$. By assumption, this tuple of polynomials does not have property (1) of Fact 4.3. We conclude that some $j\in S$ exists such that for every $k>0$, $\boldsymbol{c}(\psi_2^n\psi_3)_j=(\boldsymbol{c}(\psi_2)^n\boldsymbol{c}(\psi_3))_j>k$ for some $n\in\mathbb{N}$. Since M is parameter-reduced, $\boldsymbol{c}(\psi_1)_1$ depends on $x_{1,j}$. Hence, $\boldsymbol{c}(\psi_1\psi_2^n\psi_3)_1=\boldsymbol{c}(\psi_1)_1\boldsymbol{c}(\psi_2^n\psi_3)>k$, and $\bigsqcup M$ cannot be finite. Therefore, (1) implies (2).

Assume ϕ is an accepting computation of A. Then a tree $v\in T_\Sigma$ with $\mathrm{depth}(v)<n$ and a recursive decomposition E of ϕ exists where E is given by $\phi_o=\psi_o\tau_o(\phi_{o1},\ldots,\phi_{om_o})$, $o\in O(v)$, with $\phi=\phi_\varepsilon$ such that for every $o\in O(v)$, ψ_o is a proper (q_o, q_o)-computation for some $q_o\in Q$. If M has property (N) then for all $o\in O(v)$, $\boldsymbol{c}(\psi_o)_j$ either equals 0 or $x_{1,j'}$ for some $j'\in[1, d]$. This implies the upper bound given in (3). Since (3) trivially implies (1), this finishes the proof. □

4.2. The semiring $\boldsymbol{A}$

In this subsection, we consider again costs in the semiring $\boldsymbol{A}$. So, let $M=(A, \boldsymbol{c})$ be an $\boldsymbol{A}$-cost automaton of dimension m where $A=(Q, \Sigma, \delta, Q_F)$. Without loss of generality we assume M is parameter-reduced.

For dealing with semiring $\boldsymbol{A}$, we have to extend the notion of a monomial expression of a computation to the multidimensional case. A *monomial expression* (in short: expression) of a $(q, q_1\ldots q_k)$ computation $\phi\neq x_j$ now is a mapping $\sigma: O\to M_{\boldsymbol{R}}[X]^{[1,d]}$ with $O\subseteq O(\phi)$ such that:

(1) For $r\in O$, $\phi(r)=x_j$ implies that $\sigma(r)_i=-\infty$ or $\sigma(r)_i=x_{j,i}$. Otherwise, $\sigma(r)_i=-\infty$ or $\sigma(r)_i$ is a monomial of $\phi(r)_i$. The set of i where $\sigma(r)_i\neq-\infty$ is also called *domain* $\mathrm{dom}(\sigma(r))$ of $\sigma(r)$.

(2) $\varepsilon\in O$. If $r\in O$ and for $v\in\mathrm{dom}(\sigma(r))$, $\sigma(r)_v=h_v+\sum_{\kappa=1}^{k_v}\varepsilon_{v,\kappa}\cdot x_{j_{v,\kappa},i_{v,\kappa}}$ then $rj\in O$ iff $j=j_{v,\kappa}$ for some v and κ with $\varepsilon_{v,\kappa}\neq 0$. The domain of $\sigma(rj)$ is $\{i_{v',\kappa'}\mid j_{v',\kappa'}=j\}$.

As for dimension 1, the set of all expressions of ϕ is denoted by $\mathrm{ME}(\phi)$. The domain of an expression σ is again referred to as $O(\sigma)$. The monomial $\sigma(r)_i$ is the monomial *chosen* by σ at r and i. For $r\in O(\sigma)$ and $i\in\mathrm{dom}(\sigma(r))$, define the cost $c(r)_i$ for $r\in O$ and $i\in\mathrm{dom}(\sigma(r))$ inductively as follows: If $\phi(r)=x_j$ then $c(r)_i=x_{j,i}$ provided $i\in\mathrm{dom}(c(r))$ and $c(r)_i=-\infty$ otherwise. If $\sigma(r)_i=h_i+\sum_{\kappa=1}^{k}\varepsilon_\kappa\cdot x_{j_\kappa,i_\kappa}$ with $\varepsilon_\kappa>0$ then $c(r)_i=h_i+\sum_{\kappa=1}^{k}\varepsilon_\kappa\cdot\sigma(rj_\kappa)_{i_\kappa}$.

Finally, let $c(\sigma)_i=c(\varepsilon)_i$. Again we have

$$c(\phi)_i=\bigsqcup\{c(\sigma)_i\mid\sigma\in\mathrm{ME}(\phi)\}.$$

As for in the 1-dimensional case, monomial expressions can be composed and decomposed in correspondence to the underlying computations.

Fact 4.5. *Assume $p_1,\ldots,p_d$ are nonconstant polynomials from $A[X_1]$ such that some p_j exists for every $j'\in[1,d]$ such that p_j depends on $x_{1,j'}$. Let $b=\langle b_1,\ldots,b_d\rangle\notin\{-\infty,0\}^d$. Then either* (1) *or* (2) *is true*:

(1) *For every $j\in[1,d]$, every monomial of p_j either equals some $h\in A\setminus\{-\infty\}$ or $x_{1,j'}$ for some $j'\in[1,d]$.*

(2) *There exists some $j\in[1,d]$ such that for every $k\in A$, $p^n[b]_j>k$ for some $n\in\mathbb{N}$.*

Proof. By Fact 2.1 and induction on k, we find that for every $j\in[1,d]$, $p_j[p^{k-1}[b]]>0$. Let $m=H+\sum_{i=1}^{d}\varepsilon_i\cdot x_{1,i}$ be a monomial of p_j and $j'\in[1,d]$ such that $x_{1,j'}$ occurs in m (i.e. $\varepsilon_{j'}>0$). Then $m[p^{k-1}[b]]>p^{k-1}[b]_{j'}$ provided at least one of the following three properties hold:

(*1) Some $j_1\neq j'$ exists such that x_{1,j_1} occurs in m as well.

(*2) $\varepsilon_{j'}>1$.

(*3) $H>0$.

If for polynomial p_j, no monomial m of p_j and $j'\in[1,d]$, one of the properties (*i) holds then alternative (1) of Fact 4.5 is true. Therefore, assume that some polynomial p_j exists such that $p_{j'}$ some monomial m of p_j and some $j'\in[1,d]$ exist for which at least one of the properties (*i) holds. Since $m[p^{k-1}[b]]\leqslant p_j[p^{k-1}[b]]$, we deduce in this case that also $p_j[p^{k-1}[b]]>p^{k-1}[b]_{j'}$. Let G denote the directed graph $G=(V,E)$ with $V=\{\langle j,m\rangle\mid m\text{ monomial of }p_j\}$ and $E=\{(\langle m',j'\rangle,\langle m,j\rangle)\mid m\text{ depends on }x_{1,j'}\}$ as set of edges.

Since no polynomial is constant and every $x_{1,j}$ occurs in some $p_{j'}$, every node $\langle m,j\rangle$ is source of an edge $(\langle m,j\rangle,\langle m_0,j_0\rangle)$. It follows (similar to the case of N in the last subsection) that a strong component can be reached from every node. We mark all edges $(\langle m',j'\rangle,\langle m,j\rangle)$ in G where for m and j' one of properties (*1), (*2) or (*3) holds. Assume no strong component of G contains a marked edge. Again, it follows that every node in a strong component has indegree 1. Hence, G consists of a disjoint set cycles which means that G has no marked edge at all in contradiction to our assumption. Hence, G contains some cycle with a marked edge. This cycle has some length $r\in\mathbb{N}$. Assume $\langle m,j\rangle$ is a node on this cycle. It follows that $p^{(k+1)\cdot r}[b]_j>p^{k\cdot r}[b]_j$ for all k. Hence, $p^n[b]_j>k$ for $n=k\cdot r$. □

The d-dimensional A-cost automaton M has property (A) iff

(A) For every edge $(q, \langle \tau, j \rangle, q')$ of a strong component of the trace graph $G(A)$ and every $\mu \in [1, d]$, $c(\tau)_\mu = p \sqcup \bigsqcup_{i \in J} x_{\mu,i}$ for some $J \subseteq [1, d]$ and polynomial p which does not contain variables $x_{j,i}$ for any i.

Clearly, it can be decided in polynomial time whether or not M has property (A). We have the following theorem.

Theorem 4.6. *Assume $M = (A, c)$ is a parameter-reduced A-cost automaton of dimension d. The following three statements are equivalent:*

(1) $\bigsqcup M < \infty$,

(2) *M has property* (A),

(3) $\bigsqcup M \leqslant \begin{cases} n \cdot H & \text{if } k = 1, \\ 2 \cdot H \cdot k^n & \text{if } k > 1, \end{cases}$

where n is the number of states of A, H is an upper bound to the coefficients of polynomials occurring in $c(\delta)$ and k is an upper bound to the degree of these polynomials. It can be decided in polynomial time whether or not $\bigsqcup M$ is finite.

Proof. Assume M does not have property (A). Then there is an accepting computation $\phi = \psi_1 \psi_2 \psi_3$ for proper (f, q)-computation ψ_1, proper (q, q)-computation ψ_2 and q-computation ψ_3 such that $c(\psi_2)_j$ contains a monomial which is neither a constant nor some variable $x_{1,j'}$. Assume $S \subseteq [1, d]$ is the set of all j such that $c(\psi_2)_j \neq -\infty$. Since M is parameter-reduced, all polynomials $c(\psi_2)_j$, $j \in S$, are nonconstant, and some $j \in S$ exists for every $j' \in S$, such that $c(\psi_2)_j$ depends on $x_{1,j'}$. Therefore, we can apply Fact 4.5 to $\langle c(\psi)_j \rangle_{j \in S}$. By assumption, this tuple of polynomials does not have Property (1) of Fact 4.5. We conclude that some $j \in S$ exists such that for every $k > 0$, $c(\psi_2^n \psi_3)_j = (c(\psi_2)^n c(\psi_3))_j > k$ for some $n \in \mathbb{N}$. Since M is parameter-reduced, $c(\psi_1)_1$ depends on $x_{1,j}$. Hence, $c(\psi_1 \psi_2^n \psi_3)_1 = c(\psi_1)_1 c(\psi_2^n \psi_3) > k$; and $\bigsqcup M$ cannot be finite. Therefore, (1) implies (2).

Assume ϕ is an accepting computation of A. Then a tree $v \in T_\Sigma$ with $\mathrm{depth}(v) < n$ and a recursive decomposition of ϕ exists given by $\phi_o = \psi_o \tau_o(\phi_{o1}, \ldots, \phi_{om_o})$, $o \in O(v)$, with $\phi = \phi_\varepsilon$ such that for every $o \in O(v)$, ψ_o is a proper (q_o, q_o)-computation for some $q_o \in Q$. Assume σ is a monomial of ϕ. Then the recursive decomposition of ϕ gives rise to a recursive decomposition of σ, $\sigma_o = \sigma'_o m_o(\sigma_{o1}, \ldots, \sigma_{om_o})$, $o \in O$, where O is a suitable subset of $O(v)$ with $\varepsilon \in O$ such that $\sigma_\varepsilon = \sigma$ and $\mathrm{dom}(\sigma_\varepsilon) = \{1\}$, $c(\sigma'_o)_j = x_{1,j'}$ for some j' whenever $o \in O$, $oi \in O$ iff some x_{1,j_i} occurs in m_o, and $\sigma_{oi} = (-\infty)^d$ whenever $oi \notin O$. This implies that $c(\sigma)_1$ is bounded in accordance with statement (3). Since $c(\phi)_1 = \bigsqcup \{c(\sigma)_1 \mid \sigma \in \mathrm{ME}(\phi)\}$, this implies (3). Since (3) trivially implies (1), this finishes the proof. □

Theorems 4.4 and 4.6 give efficient versions of the result of Habel et al. [5] proving that boundedness of costs is decidable. Note that Habel et al. prove a somewhat stronger result by allowing not only polynomials over $\boldsymbol{N}$ or $\boldsymbol{A}$ but also polynomials

where all three operations $\sqcup$, $+$ and $\cdot$ occur. In fact, our methods allow to derive a polynomial decision procedure for boundedness of costs also in this slightly more general situation. It was for descriptional clarity only that we did not include a corresponding result.

5. Conclusion

We considered cost automata with cost in several semirings, derived upper bounds for bounded costs and gave polynomial-time algorithms to decide boundedness. We could not prove results in full generality for all semirings in $\{\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{F}\}$. Especially, it is rather unclear how our results on boundedness for $\boldsymbol{T}$ or $\boldsymbol{F}$ can be extended to the multidimensional case.

References

[1] B. Courcelle and M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs, *Theoret. Comput. Sci.* **109** (1993) 49–82.
[2] Chr. Ferdinand, H. Seidl and R. Wilhelm, Tree automata for code selection, in: R. Giegerich and S.L. Graham, eds., *Code Generation – Concepts, Tools, Techniques*, Workshops in Computing (Springer, Berlin, 1992) 31–50; revised version: *Acta Inform.*, to appear.
[3] F. Gecseg and M. Steinby, *Tree Automata* (Akademiai Kiado, Budapest, 1984).
[4] R. Giegerich and K. Schmal, Code selection techniques: pattern matching, tree parsing and inversion of derivors, in: *Proc. of ESOP 1988*, Lecture Notes in Computer Science, Vol. 300 (Springer, Berlin) 245–268.
[5] A. Habel, H.-J. Kreowski and W. Vogler, Decidable boundedness problems for sets of graphs by hyperedge-replacement, *Theoret. Comput. Sci.* **98** (1991) 33–62.
[6] U. Möncke and R. Wilhelm, Iterative algorithms on grammar graphs, in: *Proc. 8th Conf. on Graphtheoretic Concepts in Computer Science* (Hanser, 1982) 177–194.
[7] W. Paul, *Komplexitätstheorie* (B.G. Teubner, Stuttgart, 1978).
[8] H. Seidl, On the finite degree of ambiguity of finite tree automata, *Acta Inform.* **26** (1989) 527–542.
[9] H. Seidl, Single-valuedness of tree transducers is decidable in polynomial time, *Theoret. Comput. Sci.* **106** (1992), special issue on CAAP '90, 135–181.
[10] H. Seidl, Ambiguity and valuedness, in: M. Nivat and A. Podelski, eds., *Tree Automata and Languages* (Elsevier, Amsterdam, 1992) 355–380.
[11] H. Seidl, Finite automata with cost functions, in: *Proc. CAAP '92*, Lecture Notes in Computer Science, Vol. 581 (Springer, Berlin, 1992) 279–299.
[12] Two tree pattern matchers for code selection, in: *Proc. 2nd CCHSC Workshop 1988*, Lecture Notes in Computer Science, Vol. 371 (Springer, Berlin, 1989) 215–229.